1991

Modeling of Physical Database Design and Performance Analysis With Emphasis on VSAM Files.

Sung-eon Kim
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9207514

Modeling of physical database design and performance analysis with emphasis on VSAM files

Kim, Sung-Eon, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Modeling of Physical Database Design and Performance Analysis with Emphasis on VSAM files

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Interdepartmental Program in Business Administration

by
Sung-Eon Kim
B.S., Seoul National University, 1977
M.S., Florida State University, 1984
M.S., Louisiana State University, 1988
August 1991

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my major professor, Dr. Helmut Schneider, for his valuable guidance and support throughout the period of this research.

I would also like to thank the members of my doctoral committee, Dr. Kwei Tang, Dr. Ishwar Murthy, Dr. Ye-Sho Chen, and Dr. Bush Jones, for their helpful suggestions.

I would like to express my thanks to Dr. Tim Vaughan for his constructive help for this research before he left this school.

Most special thanks go to my wife, Gyuwon Kim, for her love, encouragement, understanding, sufferring, and patience throughout my graduate studies.

I want to thank Mr. Pil Seo for his encouragement and suggestions when I had hard times.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

Growth in the size of a database is reflected in deterioration of database performance. Since deterioration is related to the structure of the file, the performance efficiency involves the design of a physical database and the proper management of it.

This research addresses a modeling procedure of a physical database design considering both records insertions and deletions. The model describes the behavior of a physical database in a VSAM file environment, and is extended to the issue of database reorganization through a cost analysis. The cost of accessing the database increases because of the physical disorganization of the database caused by records updates and insertions. A cost function that describes this excess cost is defined. As a remedy to the performance deterioration, database reorganization is required. Optimum reorganization points are obtained as a tradeoff between the excessive access costs and the reorganization cost. Numerical examples based on the characteristics of IBM 3380 are given.

Chapter 1

INTRODUCTION

The complexity of every aspect of modern society has greatly increased the demand for information of all kinds. When the volume of information becomes too huge to be handled by hand, organizations turn to computers to process the data efficiently. Computers store information as records in a file. The structure of this file deteriorates over time as the number of transactions increases. These transactions include, for example, insertions, deletions, retrievals, and updates. Guarding against file deterioration becomes an increasingly greater challenge for the database administrator. Since deterioration is related to the structure or organization of the file, the solution involves the design of a physical database and the proper management of files through reorganization (Koushik, 1987). A physical database is a collection of interrelated data that are stored together as one or more types of records. Since files can be considered degenerated forms of physical databases in which the number of stored record types is one, the terms physical database and file can be used interchangeably when the stored record type is one.

The physical database design problem centers on the choice of an efficient means for implementing the database. To solve this problem, one must make decisions concerning

file organization and the design parameters to be used for that organization. A file organization is a representation of the stored records that make up a file. The representation shows the record format, logical and physical ordering, potential access paths such as indexes, and physical device allocation. Two main file organizations are hashed files and indexed-sequential files. The latter can be subdivided into indexed-sequential access method files and virtual storage access method files. The choice of a file organization may depend on anticipated use of the database. Values of the design parameters depend on the associated file organization and should be determined precisely. Design parameters include, for example, initial loading factor, block capacity, overflow handling method, and capacity of a file.

With regard to file reorganization, the problem is to determine the optimal time required to reorganize the file physically. Reorganization is the process of unloading (reading) records from a file and reloading (writing) these records on a restructured file. The optimal reorganization point depends largely on the file organization chosen to support the database and on the frequency of transactions on this database. The reason is that both the overflow problem and the implementation of deletion are handled differently for each file organization. A modeling structure can be changed according to the implementation of deletion. If

implementation is by logical deletion, the garbage collection effect appears at the time of reorganization. Logical deletion means that deleted records are marked at the deletion time and are physically deleted at the time of reorganization only. In a physical deletion implementation, the deleted record space is immediately reusable; thus, the frequency of reorganization is less than that in a logical deletion implementation.

Frequent reorganizations are not desirable because reorganization is a lengthy and costly procedure. In general, the reorganization point can be determined by comparing total excess cost and reorganization cost so as to minimize total operating cost. However, sometimes it is suggested to reorganize the database at a fixed time or at the end of the n^{th} time period after the most recent reorganization.

1.1. Statement of the Problem

Problems relating to physical database design and performance analysis have been studied over the last decade. To analyze these problems, researchers have adopted stochastic and analytic modelings. In many researches that focus on database reorganization, it is assumed that an efficient physical database design already exists or that characteristics of the database deterioration process have

already been given. Since deterioration is influenced by the physical database design that is chosen, it may be desirable to treat them jointly.

The main objectives of this research are to integrate physical database design problem and reorganization problem and to develop a combined model of these problems.

1.2. Research Framework

The research effort addresses the modeling procedure of physical design for growing databases in the environment of a virtual storage access method (VSAM) file.

Unlike a hashed file and an indexed-sequential access method (ISAM) file, a VSAM file does not have an overflow area. Instead, it handles the overflow problem through control interval (CI) splits and control area (CA) splits. A CI is the unit of information that the VSAM file transfers between virtual storage and disk storage.

With the increase of records in a database, CIs are filled up. When this happens, a CI split occurs at the next record insertion. In this event, half the records remain in that CI and the other half move to one of the CIs that are allocated as free space in a CA at the database loading time. A CA consists of a fixed number of CIs, and the maximum size of a CA is a cylinder. As a result of continuous CI splits, if free space is no longer available

in a CA, a CA split occurs in a manner similar to a CI split. When a CA split occurs, a new CA is established at the end of the data set and all the records in half the CIs of the old CA move into a new CA. With CA splits, sequential processing slows because of the increase in seek time (Ranade, 1987). In other words, growth in the size of a database is reflected in the deterioration of performance for that database. This performance deterioration causes an increase in record search costs. Therefore, to restore performance efficiency and to reduce the search cost, one must turn to database reorganization. As reorganization also requires extra cost, however, optimal reorganization policy is set up in the way of minimizing the total operation cost. The optimal reorganization point is calculated based on the number of CIs or CAs obtained from the physical database design model.

1.3. Contribution of the Research

The analysis of the performance of database file structure has been addressed by many researchers over the last decade. Most of their works have focused on the hashed files or ISAM files. Only a few studies have centered on the VSAM files (Maruyama and Smith, 1976; Chin, 1978). Even in these studies, the physical database design and reorganization problems were not analyzed as a combined

problem, and any specific model for the behavior of the physical database was not provided.

The present research contributes to the current knowledge in handling physical database design and reorganization problems in the following ways. First, this research attempts to model the physical database design in a VSAM file for the first time in the literature. The model deals with two difficult factors that the VSAM file adopts. One is the physical record deletion, and the other is CI and CA splits that are used as a means of overflow handling. Second, this research attempts to combine the physical database design problem and the reorganization problem.

1.4. Organization of the Research

This research is organized into eight chapters. In Chapter 2, the structures of files such as hashed, ISAM, and VSAM are described to provide a basic difference among files. The literature review in Chapter 3 focuses on prior researches on physical database design and reorganization problems. In Chapter 4, models of the physical database design for a VSAM file are described for the following situations: (1) when only continuous records insertions are considered; and (2) when both records insertions and deletions are considered. In Chapter 5, numerical analyses are performed for the models developed in Chapter 4 using

the various parameter values and the results are discussed. The second model developed in Chapter 4 is used in Chapter 6 to do the performance analysis. Through this performance analysis, the way to determine the optimal reorganization policy is discussed. File structure at each reorganization period is updated through the time of reorganization. File operation and reorganization costs are obtained to determine the optimal reorganization point. In Chapter 7, IBM 3380 environment is introduced to discuss the optimal reorganization policy obtained in Chapter 6 numerically. The final chapter gives the summary and conclusion of the research.

Chapter 2

DESCRIPTION OF FILE STRUCTURE

A database system includes files and the programs to manage those files. A Database Management System (DBMS) helps to manage the related data in multiple files. A DBMS uses file management services to manipulate many files in the system. A database is defined as a collection of related data. The data storage for a database is accomplished through the use of one or more files. A file is a collection of records kept on computer storage devices. A file will have a name and a structure that is determined by a file access program. A record is a collection of related fields containing elemental data items.

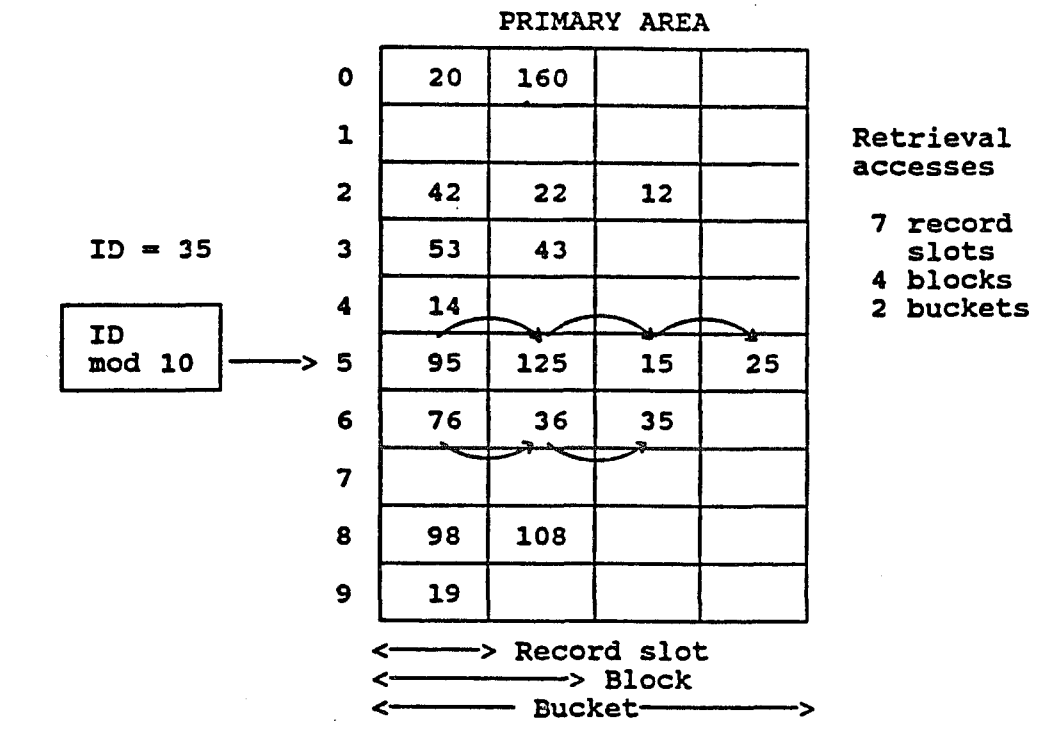
Given the position of the file, we can locate the records and update their fields. The performance of a database system can vary greatly, however, depending on the file organizations when records are stored and retrieved. Two file structures widely used in the performance analysis are the hashed file and the indexed-sequential file. These files are supported by major commercial DBMS's such as IDMS, IMS and TOTAL (Park et al, 1988). These files have similarities in the way that they incorporate growth in the number of records over time.

2.1. Hashed Files

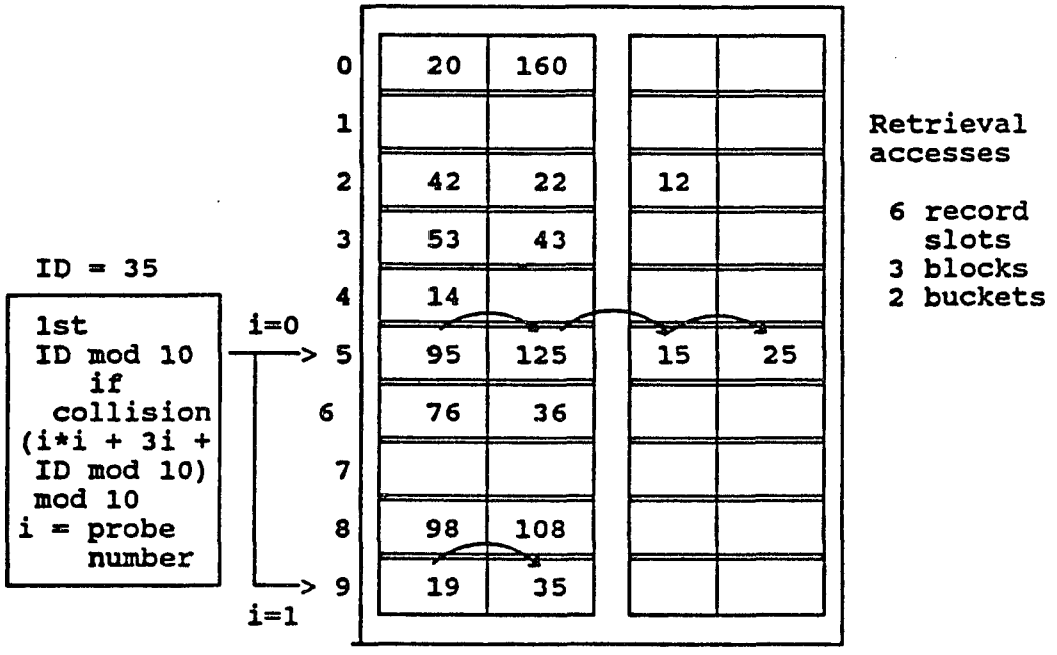
Hashed files are based on direct access to a file using a relative address. The relative address is an integer ranging in value from zero to the maximum number of blocks within the domain of the system which controls the storage of data. The relative address of a record is determined by applying a hashing function to the record key. Such a key-to-address transformation (also commonly referred to as a randomizing hashing function) translates the key attribute values into relative addresses within the file space and gives each arriving record its own slot, based on the key provided. One problem with the key-to-address transformation is the possibility of a collision. A collision occurs when identical addresses are generated from different source of keys by a hashing function so that more than one record is directed to the same place in a storage. If a collision occurs when all record slots for a given bucket are filled, then an overflow occurs. Selection of overflow handling techniques is one of the most important decisions in the design of a random access method. The basic types of overflow techniques are Open overflow, Nonlinear search, Coalesced chaining, and Separate chaining (Severance and Duhne, 1976; Teorey and Fry, 1982). The Open overflow method stores an overflow record in the first unused or open record slot in the next unfilled bucket. In the nonlinear

search method, rehashing within a sequence of identifier transformation functions is used to find an open slot to either store a new record or retrieve that record later. Coalesced chaining provides for a group of unused buckets at initial load to allow for overflow. The Separate chaining method allocates local overflow record buckets for each home address bucket. This method is similar to the overflow chains used in indexed-sequential files except that records are not maintained in any sequential order. These overflow techniques are illustrated in Figure 2.1.

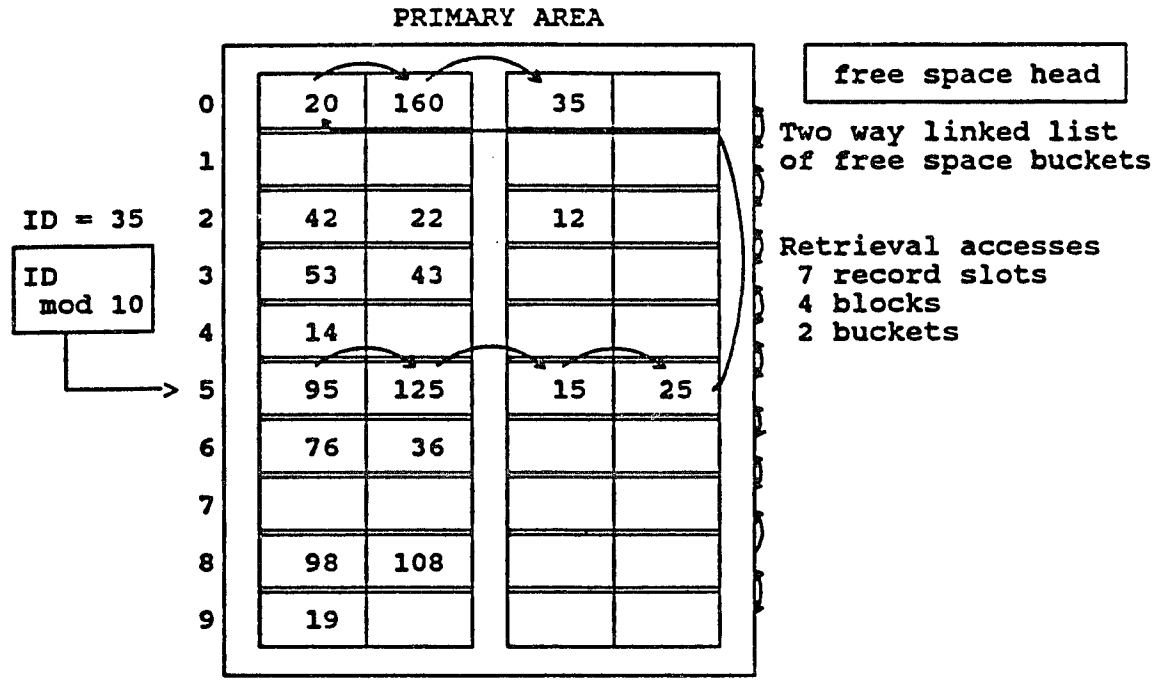
Figure 2.1 Hashing overflow techniques (Teorey and Fry, 1982)



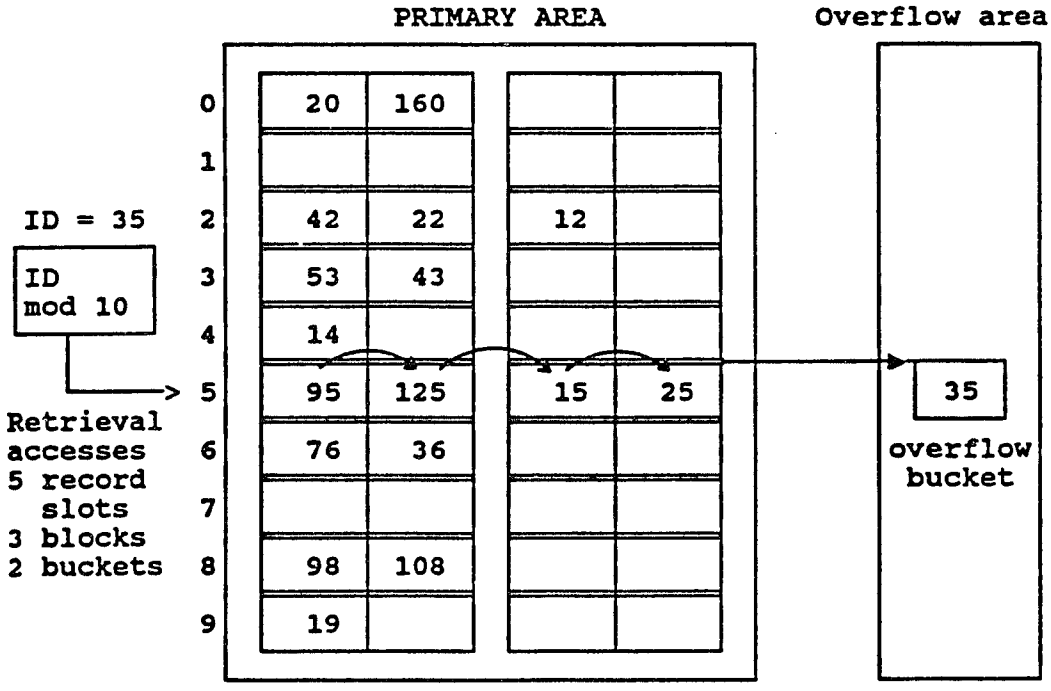
(a) Open overflow



(b) Nonlinear search



(c) Coalesced chaining



(d) Separating chaining.

2.2. Indexed-Sequential Files

Indexed-sequential file design attempts to overcome the access problem inherent in sequential file organization without losing all the benefits and tradition associated with sequential files. The indexed-sequential file has two additional features beyond the organization of a sequential file. One is an index to provide better random access. The other is an overflow area to provide a means for handling additions of records to the file. This file uses two types of access. One is sequential, and the other is random. The ratio of the amount of sequential access to random access in the usage may affect the choice of this file. However, because the file permits both sequential and random access to the data records in the database, it is a frequent choice in database applications.

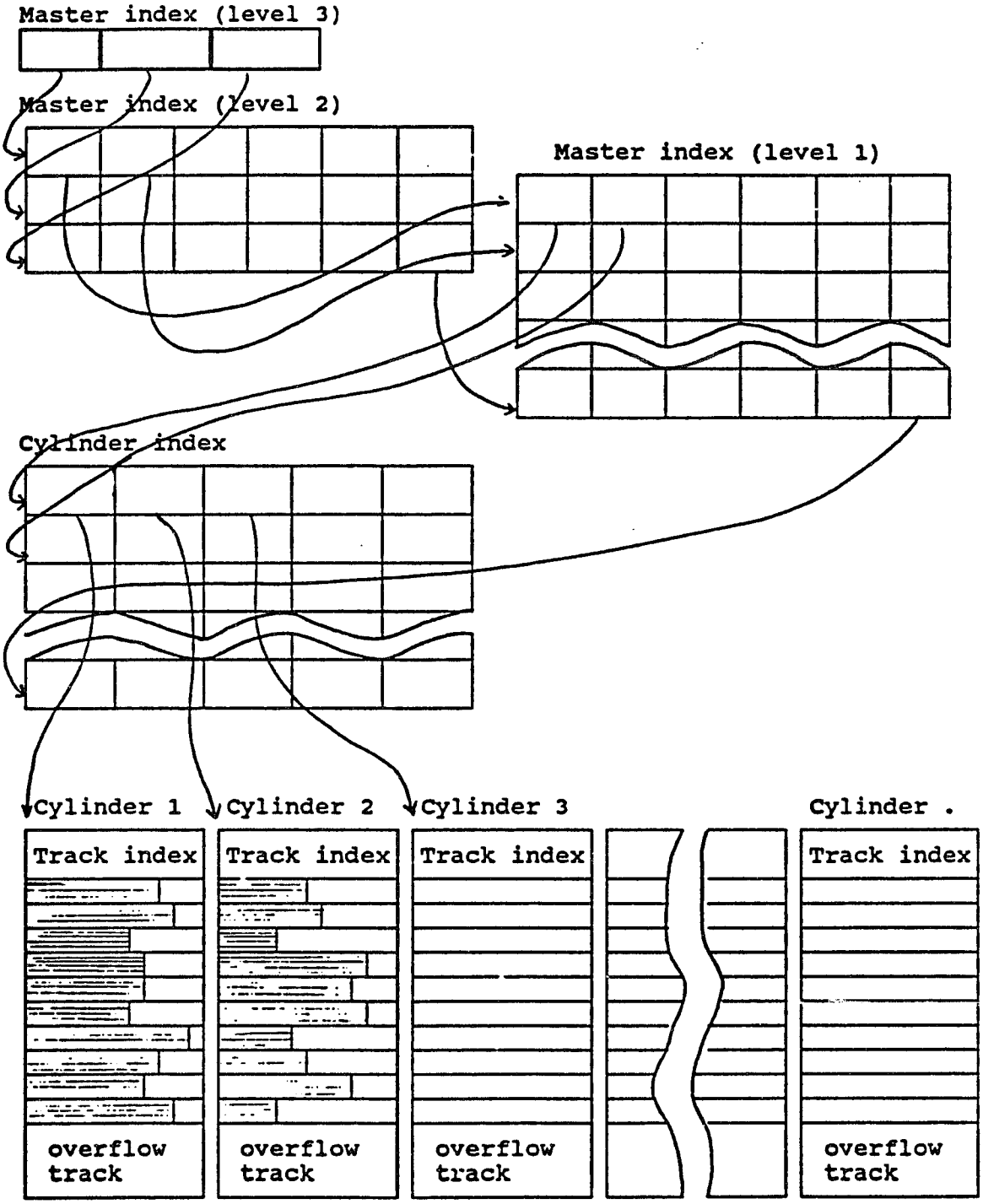
The indexed-sequential file can be made to fit a specific file hardware configuration. This means that the type of storage devices may be changed or that the file may be moved from one type of storage device to another. In this file, two access methods used within the IBM System 370 can be considered. One is ISAM (indexed-sequential access method) in which the indexes and blocks are designed to fit specific file units. The other is VSAM (virtual storage access method) which is hardware-independent.

2.2.1 ISAM Files

An ISAM file consists of an ordered physical sequential file and a hierarchy of track indexes, each ordered by primary key values in the same way in which the data file is ordered. In ISAM files, the records are grouped to fit onto physical disk tracks, and one track on each cylinder contains an index to the records stored in that cylinder. When the file is initially created, all the records are located in primary storage areas. If new records are inserted when the original sequential file has been filled up, they are stored in an overflow area. The index track contains pointers both to the prime data area and to the overflow area.

An ISAM file has three levels of indices. The lowest level of index is the track index, which contains the highest-value key on each track and points to that track. The cylinder index is the next higher level index, which contains highest-value key on each cylinder and points to the track index of that cylinder. The highest level of index is called the master index. It contains the highest-value key on each track of the cylinder index, along with a pointer to that track. Figure 2.2 shows the basic structure of a ISAM file. The way that an ISAM file addresses the overflow is to use pointers from the track index and then chains to indicate the key sequence of the inserted records.

Figure 2.2 Basic view of a ISAM file.



When a record is inserted, it is fitted into prime data track. If there is no room in a track for a new record that has been inserted, then it is written on the overflow track.

2.2.2 VSAM Files

The VSAM has been developed for use with virtual storage operation systems. It grows out of the need for an access method that allows data to be accessed both directly by key and sequentially in key-defined collating order (Keehn and Lacy, 1974). The way a VSAM file reads a record is similar to that of an ISAM file except that the operation is not described in terms of track and cylinders. In a VSAM file, instead of cylinders that are subdivided into tracks, control areas are subdivided into control intervals (Martin, 1975).

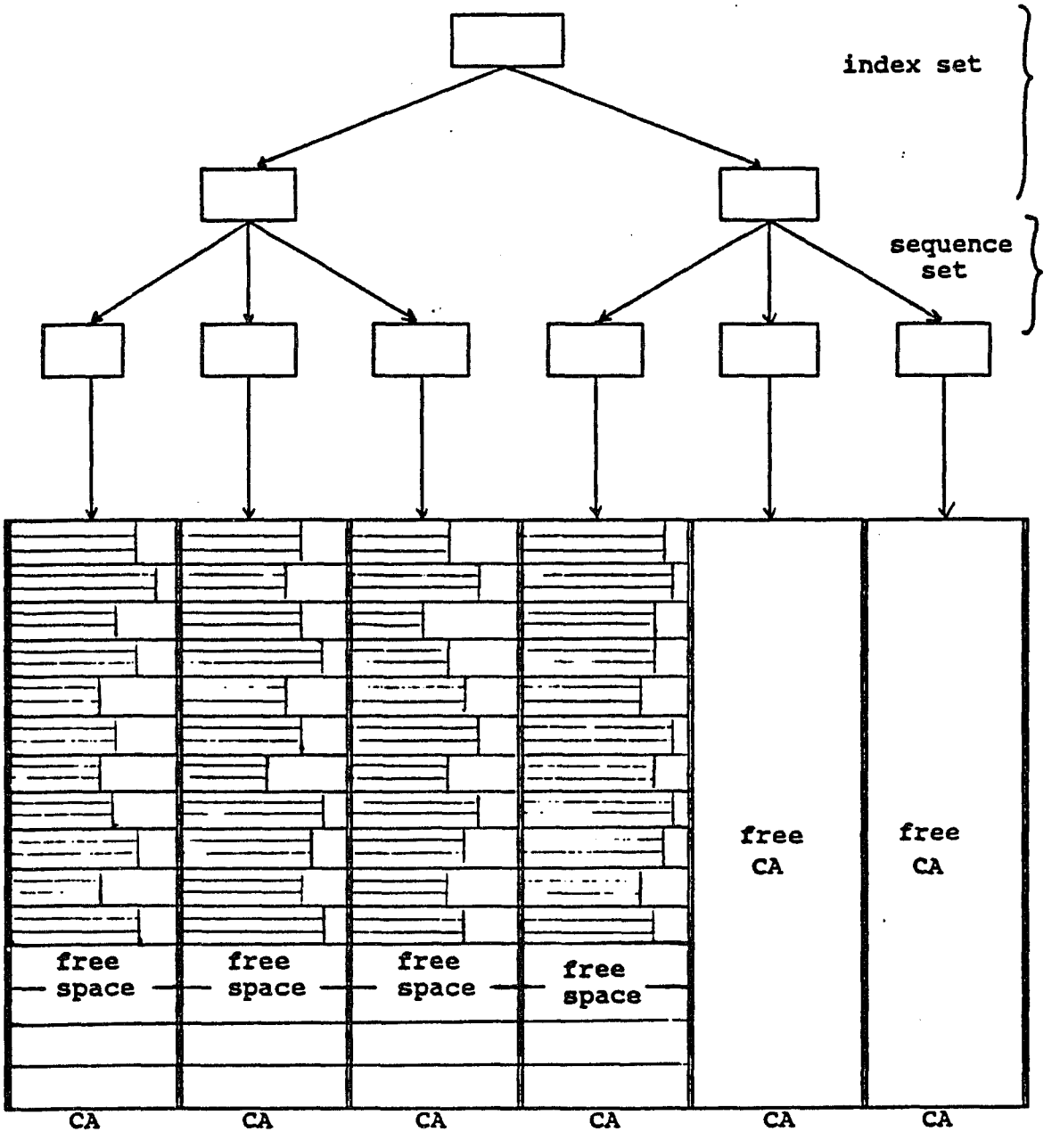
A control interval (CI) is a continuous area of direct access storage that a VSAM file uses to store data records and to control information that describes these records. A CI is the unit of information that a VSAM file transfers between virtual storage and disk storage. Whenever a record is retrieved from direct access storage, the entire CI containing the record is read into a VSAM file I/O buffer in virtual storage. The CIs in a VSAM file data set are grouped together into fixed-length contiguous areas of direct access storage called control areas (CAs). A VSAM

file fixes the number of CIs for each CA in a file. The maximum size of a CA is one cylinder, and the minimum size is one track of Direct Access Storage Device (DASD) storage. The basic view of a VSAM file is given in Figure 2.3.

In a VSAM file, unused space can be scattered throughout the data set as free space. The space that was occupied by the deleted record is available as free space, because when a record is deleted, the record is physically erased. Insertions into a key-sequenced data set use the free space provided during the definition of the data set, provided by the record deletions, or developed as a result of CI and CA splits.

CI splits and CA splits are particular ways of handling the overflow problem in a VSAM file. As was discussed in Section 1.2, if there is not enough free space in a CI to accommodate the new records, then a CI split takes place at the point of insertion. CI split causes the data CIs to have a physical order that differs from the key sequence. When a CI split occurs, the records comprising the first half of the full CI are copied into the new CI and the old CI is adjusted by shifting the latter half of the records to the front. The index is updated to include the new CI at the appropriate point. The last record provides the key value for the index entry (Wiederhold, 1987). When free space is no longer available in a CA, a split occurs in a manner similar to a CI split. When a CA split occurs, a

Figure 2.3 Basic view of a VSAM file



new CA is established at the end of the data set and half of the CIs with all their data records moves into a new CA. With CA splits, sequential processing becomes slow because of the seek time increase. While the use of overflow areas in a ISAM file always increases the access time needed to read the inserted records, the use of distributed free space in a VSAM file generally results in higher storage requirements than an overflow chaining method. However, it permits much faster retrievals of inserted records (Martin, 1975).

The index structure of a VSAM file is also different from that of an ISAM file. While an ISAM file has a track index and a cylinder index, a VSAM file has a CI index and one sequence set index per CA. The sequence set index is itself indexed by a hierarchy of indices called index set. The sequence set index contains the highest key value in each CI and points to that CI. The lowest level of the index set contains the highest-value key in each CA and points to the sequence set index block for that CA. The structure of the VSAM file index blocks is implemented by B^+-trees (Comer, 1979). In a B^+-tree, all keys reside in the leaves. The upper levels consist only of an index and are organized as a B-tree.

Chapter 3

LITERATURE REVIEW

Many authors have contributed to the body of researches concerning file design and reorganization. Classification of the published literature gives a perspective on the relation of the studies and provides a means for comparing related efforts. Classification could be done by the methodologies the authors adopted. To identify the specific approach used to analyze and to solve the given problem, the methodology can be broken down into following two types: stochastic and analytic/heuristic. Classification could also be done by the file organizations adopted. However, since the applicability of some models may not be limited to one file structure, this classification may not be desirable. Simulation-oriented early works are not reviewed because of the differences in the approach method expected in the present research.

3.1. File/Physical Database Design and Reorganization: Stochastic Approach

Van der Pool (1973) adopted the storage allocation of hash-based files in steady state. The evolution of the file is modeled as a Markov chain. It was assumed that the insertion characteristic follows a homogeneous Poisson

process and that life times are distributed exponentially. The loading factors that minimize file maintenance costs in terms of storage space and additional accesses are computed for different bucket sizes and different operational conditions.

Larson (1981) analyzed the performance deterioration of indexed-sequential files under the assumption that overflow records are handled by chaining. The objective of this research was to quantify the effects of insertions and deletions (1) on the number of overflow records, and (2) on the number of additional accesses for both successful and unsuccessful retrieval performance. He modeled the dynamics of the file system as an instance of a birth and death process. The insertion characteristic was modeled by a homogeneous Poisson process, while the record life times were taken to be exponentially distributed. In the development of the model, a simplifying assumption was made about the distribution of the key interval assigned to a bucket.

Under the assumption of incremental reorganization, the distribution of overflow records was determined and the steady-state was rapidly achieved for a stable file. He considered the file growth problem explicitly, but did not formulate the file reorganization problem.

Heyman (1982) modeled the performance deterioration by diffusion models from which the number of overflows from a

block is estimated. As Larson (1981) assumed, the deletion rate was assumed to be constant and was treated to be independent of the number of records in the bucket. He examined a formula for the calculation of the expected number of overflows by time. He derived a simple formula, different from Larson's, for the optimal time to reorganize a stochastically growing database by using renewal theorem.

Cooper and Solomon (1984) extended Larson's work to consider the problem of the average time until bucket overflow. The growth of the file was modeled as a renewal process. The inter-arrival time between records was assumed i.i.d. random variables of arbitrary distribution. However, their model cannot be generalized to handle residence time with an arbitrary distribution while Larson's can.

Their work was also compatible with Heyman's (1982) work although there are some differences. One difference is the assumption for the deletion rate. They assumed that the deletion rate is proportional to the number of records in the bucket. The other difference is the fundamental calculation of the average time until the bucket first overflows. A result showed that the average time until bucket overflow is greatly reduced by increasing the bucket capacity for the same load factor. This model was created for hashed files and has a limitation to be applied to indexed-sequential files.

Mendelson and Yechiali (1981) considered a database where record additions are governed by a renewal process and record deletion is a logical deletion. In logical deletion, deleted records are flagged and physically deleted when a reorganization occurs. They modeled the reorganization problem as a semi-Markov-decision problem and proved that the optimal state-dependent decision rule is a deterministic control-limit rule. This rule calls for a reorganization as soon as the number of records in the system reaches or exceeds a fixed control-limit. They developed an optimal control-limit evaluation procedure. In this procedure, reorganization occurs as soon as the expected one-stage cost rate per unit time resulting addition of a record exceeds the expected average total cost of the system per unit time. They concentrated on proving that the "cost-cutoff" policies considered by Shneiderman (1973) are optimal without a loss of generality.

Koushik (1987) considered the combined problem of a file design and a reorganization. The major design parameters involved are the blocking factor and the loading factor. The problem of a file reorganization is formulated as a stochastic dynamic program. Models for both hashed files and indexed-sequential files have been presented.

Park et al. (1988) presented a theoretical approach to analyze the optimal file reorganization policies using a dynamic programming method. They integrated the micro-level

stochastic model of file state dynamics and the cost structures of operations that are derived from a low-level analysis of the physical file design. They also developed solution procedures such as an integrated model for evolutionary and stationary databases. In setting optimal file reorganization policies, they considered a reorganization only when the arriving transaction was an insertion under the assumption of logical deletion of records. The model and solution procedures were applied to an indexed-sequential file.

3.2. File/Physical Database Design and Reorganization: Analytic/Heuristic approach

Shneiderman (1973) introduced the problem and solution techniques with a few typical cases for a database reorganization. In part I of his study, he considered a deterministic steady-state model with linearly increasing search and reorganization costs and a finite database lifetime. He found the optimal fixed-length reorganization point, assuming fixed-length time intervals between reorganization instants. In part II, he considered the case of random deterioration, where the search cost was a stochastic process with a given time-dependent distribution. He considered two reorganization strategies: reorganization at fixed time intervals and reorganization when the search cost has determined to a given level.

Yao et al. (1976) studied the reorganization problem for five file reorganizations: direct, sequential, indexed-sequential, multi-list and inverted file structures. They argued that, since the lifetime of the database is unknown, and because of computational efficiency considerations, it is preferable to use a heuristic dynamic reorganization algorithm. This algorithm calls for a reorganization as soon as the search cost savings due to a reorganization become at least the average reorganization cost per time period. They showed that this algorithm yields near-optimal results for the deterministic case with linearly increasing costs. When costs are fixed, it is shown that, fixed-length reorganization intervals are optimal, while increasing costs result in increasing reorganization intervals. This algorithm was also applied to various file reorganization schemes with nonlinear search costs.

Tuel (1978) extended Shneiderman's (1973) work on linearly growing files and obtained the optimal solution when variable-length reorganization intervals are allowed.

Maruyama and Smith (1976) considered databases that were stored on disk devices using a disk file organization which allowed free space to be distributed throughout the file. The insertion characteristic was modeled as a non-homogeneous Poisson process, and the insertion rate was regarded as the difference between the actual insertion and

deletion rates. However, deletions were not explicitly considered.

They found that the physical disorganization of the disk file of this type is nonlinear, and that the impact of the disorganization on accesses to the file depends on the sequentiality of the accesses. They showed that the optimum reorganization point also depends on the sequentiality of accesses to the file and concluded that for some files the optimum strategy is not to do a reorganization.

Chin (1978) studied the size of an expected distributed free space within a data storage area (DSA) for ordered and non-ordered access methods in terms of number of records. When a file is created using an ordered access method, records in a DSA are stored, maintained, and retrieved with respect to a preordered sequence that depends on the value of a selected attribute. When a file is created using a non-ordered access method, records within a DSA are stored, retrieved, and maintained in a random, non-ordered sequence. He described the optimal block size for a DSA, which minimizes the CPU operations and I/O interruptions, under the following assumptions: (1) the CPU cost is proportional to the size of a physical block, and (2) the I/O cost is proportional to the number of blocks.

Leung (1986) studied the time-dependent deterioration and performance degradation of files. He considered both the insertion and deletion characteristics. However, he

assumed that the records are deleted logically and that they are deleted physically at a reorganization time. He established a connection between the dynamic fragmentation pattern and the concrete microscopic parameters of record insertion and deletion. Since the actual file performance behavior is difficult to observe, such a connection is useful for prediction purposes. Deterioration characteristics are presented for non-homogeneous Poisson insertion process and general record lifetime distribution. He obtained a closed-form expression for the optimal compaction interval for the case of constant record insertion rate and exponential record lifetime distribution.

Chapter 4

MODELING OF PHYSICAL DATABASE DESIGN

This chapter describes the modeling of the physical database design for a VSAM file. The general descriptions and the operations on VSAM are well represented in many books (Ranade and Ranade, 1986; Ranade, 1987; IBM Corp., 1985). VSAM is IBM's strategic access method for MVS/SP, VS/XA, DOS/VSE, OS/VSI, and VM/CMS systems. VSAM supports three different data set formats: key-sequenced, entry-sequenced, and relative record. In this research, a key-sequenced data set (KSDS) is used as the underlying file structure because it fulfills the same functional requirements as ISAM. A KSDS consists of two physical components on the direct access storage device. The first of these is the index component, which contains the key fields and pointers to the location of the record to which that key field belongs. The second is the data component, which contains the records that hold the user data including the key field. The key field is small compared to the whole record; similarly, the index component is small compared to the data component. Therefore, the main concern of this research is the data component. In a KSDS, logical records are placed in the data set in ascending collating sequence by key field. Records can be retrieved and inserted, both randomly and sequentially. When a KSDS is created, unused

space, called free space, can be scattered throughout the data set to allow records to be inserted. Free space within a CI is used for in-place reorganization of KSDS's for additions, updates, or deletions of records within that CI (IBM Corp., 1985). To some extent, this helps to keep the data components in physical sequence in spite of many subsequent random insertions. However, as a result of excessive random insertions and deletions, the components eventually become physically out of sequence, although they are still in logical sequence when accessed through the index of the cluster (Ranade and Ranade, 1986).

The main purpose of this chapter is to develop models that describe such a physical database deterioration by analyzing the expected behavior of a physical database structure in which splits are used to handle overflows. The difficulty of handling the model comes from the fact that the probabilities of inserting a record into each CI changes continuously with the change of total records in the system. Another difficulty in modeling comes from a CI split. Since the split changes the state of a CI, it is difficult to define any distribution function that describes the number of records in each CI, differently from the analytical method used in ISAM file analysis. In developing a model, it is assumed that no record spans over CI boundary are allowed, and that the data record size is fixed. These

assumptions will be applied both in the following two sections.

4.1. Modeling of a Physical Database When Continuous Insertions Are Considered

The modeling described in this section considers successive random insertions of records in the construction of a physical database. The assumptions given in this section are (1) that records are uniformly distributed according to keys of records, and (2) that insertion is at random. If there are n keys in a database, these n keys divide all possible key values into n+1 slots. If an insertion has an equal probability of being in any one of these n+1 slots, it is called a random insertion and its probability is given by 1/(n+1). However, if we let f(i) be i/(n+1), the total sum of f(i), i = 1, ... n, becomes n/(n+1), which is not one. This situation can be relieved in the following way. We know that if a CI has i records, it provides i slots for a newly inserted record except for either the first CI or the last CI, which has one more slot than the others. The generalization of the fact that, if there are i records, there are i slots, can be obtained by putting a dummy key at the beginning of the first front CI. If the system has a large enough number of records, the effect of this dummy key will be insignificant (Baeza-Yates,

1989a). The analysis of this model is based on the fringe analysis. Fringe analysis has been used to analyze files whose indices have tree structures such as 2-3 trees, B-trees, and B^+-trees (Yao, 1978; Eisenbarth et al., 1982; Baeza-Yates, 1989a, 1989b). The research stream of these researchers was to find the expected behavior of balanced search trees. The composition of a fringe is described in several ways. Yao (1978) described it as the expected number of trees of each different type of 2-3 trees. Eisenbarth et al. (1982) indicated that the fringe of a tree consists of one or more subtrees that are isomorphic to members of a tree collection. Typically, the fringe contains all subtrees that meet this description. The description of the fringe is modified in this research, however, to the expected number of CIs that contain i records after the n^{th} insertions, which is denoted by $X_i(n)$.

The transitions between CIs can be used to model the insertion process. When a record arrives for an insertion into the database, it is assigned to a CI. The method used for determining the assignment depends on the file organization technique. In an indexed-sequential file such as VSAM, the assignment is made by means of an index look-up. If the assigned CI is not full when an insertion occurs, then a new record is added to that CI and the insertion process is completed. If the assigned CI is full, then a CI split occurs. In the former case, the count of

the number of records in the assigned CI is increased by one. In the latter case, as was discussed in Section 1.2, because of the split, the count of the number of records in the assigned CI is decreased to $(b+1)/2^*$ and a new CI with $(b+1)/2$ records is created.

Let the CI that contains i records be denoted by CI^i. The insertion of a record into a CI^i will cause a decrease in the number of CI^is by one and an increase in the number of CI^{i+1}s by one, because a CI^i will be converted to a CI^{i+1} with the addition of a record. Since the current value of X(n) depends on the history of the process only through the recent value of X(n-1), this insertion process follows a Markovian. According to Hillier and Lieberman (1986), a sequence $\{ X_{(n)} \} = \{ X(0), X(1), \ldots\}$ of random variables taking values on a set space S is a Markov chain if

$$\Pr \{ X(n) = i \mid X(n-1) = j, \ldots, X(0) = k \}$$
$$= \Pr \{ X(n) = i \mid X(n-1) = j \} \text{ for all } i, j, \ldots, k \in S.$$

Recall that $X_i(n)$ is the number of CI^is when there are n records in the file. The probability that a record is inserted into a CI^i is assumed to be $iX_i(n)/(n+1)$. With this probability, $X_i(n)$ is decreased by one and $X_{i+1}(n)$ is increased by one, because one record insertion into a CI^i

---

* b is the capacity of a CI and is assumed to be an odd number for analytical convenience. Therefore, (b+1)/2 will be an integer number.

will make it a CI^{i+1} as explained earlier. Similarly, the above relationship also holds for $X_{i-1}(n)$ and $X_i(n)$ with probability $(i-1)X_{i-1}(n)/(n+1)$. These relationships are represented by the following transition probabilities when a record is inserted:

$$\Pr\{ X_i(n+1) = a_2 \mid X_i(n) = a_1,\ X_{i-1}(n) \}$$

$$= \frac{iX_i(n)}{n+1} \qquad \text{for } a2 = a1 - 1 \qquad (4.1.1)$$

$$= \frac{(i-1)X_{i-1}(n)}{n+1} \qquad \text{for } a2 = a1 + 1 \qquad (4.1.2)$$

$$= 1 - (4.1.1) - (4.1.2) \qquad \text{for } a2 = a1, \qquad (4.1.3)$$

where $i = 2,3, \ldots, b$, and b is the maximum number of records that a CI can hold; i.e., capacity of a CI. When $i = 1$, some of the above transition probabilities are changed by the following equations:

$$\Pr\{ X_1(n+1) = a_2 \mid X_1(n) = a_1 \}$$

$$= \frac{X_1(n)}{n+1} \qquad \text{for } a2 = a1 - 1 \qquad (4.1.4)$$

$$= 1 - (4.1.4) \qquad \text{for } a2 = a1. \qquad (4.1.5)$$

If an insertion falls on a CI^b, with probability $bX_b(n)/(n+1)$, a CI split occurs. The effect of a CI split could be combined on the transition probabilities for the CIs with k1 or k2 records. k1 and k2 are defined as $k1 = \lfloor (b+1)/2 \rfloor$, $k2 = \lceil (b+1)/2 \rceil$, where $\lfloor (b+1)/2 \rfloor$ denotes the

integer part of (b+1)/2 and $\lceil (b+1)/2 \rceil$ denotes the smallest integer $\geq$ (b+1)/2. To make the analysis simple, let us define integer value k = (b+1)/2 by selecting only odd numbers for b. Since a CI split produces two CI^ks, the transition probabilities reflecting this CI split are given by the following equations:

$$\Pr \{ X_k(n+1) = a_2 \mid X_k(n) = a_1, X_{k-1}(n), X_b(n) \}$$

$$= \frac{kX_k(n)}{n+1} \qquad \text{for } a2 = a1 - 1 \qquad (4.1.6)$$

$$= \frac{(k-1)X_{k-1}(n)}{n+1} \qquad \text{for } a2 = a1 + 1 \qquad (4.1.7)$$

$$= \frac{bX_b(n)}{n+1} \qquad \text{for } a2 = a1 + 2 \qquad (4.1.8)$$

$$= 1 - (4.1.6) - (4.1.7) - (4.1.8) \quad \text{for } a2 = a1 \qquad (4.1.9)$$

Using the above transition probability equations (4.1.1) through (4.1.9), one can obtain the expected values of $X_i(n+1)$ conditional on $X_i(n)$ and $X_{i-1}(n)$ by:

i) when i = 2, 3, ..., b, $\neq$ k

$$E [X_i(n+1) \mid X_i(n), X_{i-1}(n)]$$

$$= (X_i(n) - 1) \frac{iX_i(n)}{n+1} + (X_i(n) + 1) \frac{(i-1)X_{i-1}(n)}{n+1}$$

$$+ X_i(n) \left(1 - \frac{iX_i(n)}{n+1} - \frac{(i-1)X_{i-1}(n)}{n+1}\right)$$

$$= \left(1 - \frac{i}{n+1} \right) X_i(n) + \frac{(i-1)X_{i-1}(n)}{n+1} \qquad (4.1.10)$$

ii) when i = 1

$$E\,[X_1(n+1) \mid X_1(n)] = \left(1 - \frac{1}{n+1} \right) X_1(n) \qquad (4.1.11)$$

iii) when i = k

$$E\,[X_k(n+1) \mid X_k(n),\ X_{k-1}(n),\ X_b(n)]$$

$$= (X_k(n) - 1)\,\frac{kX_k(n)}{n+1} + (X_k(n) + 1)\,\frac{(k-1)X_{k-1}(n)}{n+1}$$

$$+ (X_k(n) + 2)\,\frac{bX_b(n)}{n+1}$$

$$+ X_k(n)\,\left(1 - \frac{kX_k(n)}{n+1} - \frac{(k-1)X_{k-1}(n)}{n+1} - \frac{bX_b(n)}{n+1}\right)$$

$$= \left(1 - \frac{k}{n+1} \right) X_k(n) + \frac{(k-1)X_{k-1}(n)}{n+1} + \frac{2bX_b(n)}{n+1} \qquad (4.1.12)$$

Now, we wish to obtain the unconditional expected value of $X_i(n+1)$. This can be done in the following way (Ross, 1984):

$$E\,[\,E\,[X_i(n+1) \mid X_i(n),\ X_{i-1}(n)]\,]$$

$$= \Sigma_{\alpha 1}\Sigma_{\beta}\ E\,[X_i(n+1)=\alpha 2 \mid X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta]$$

$$\times\ P\,\{X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}$$

$$= \Sigma_{\alpha 2}\Sigma_{\alpha 1}\Sigma_{\beta}\ \alpha 2\ P\,\{X_i(n+1)=\alpha 2 \mid X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}$$

$$\times\ P\,\{X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}$$

$$= \Sigma_{\alpha 2}\Sigma_{\alpha 1}\Sigma_{\beta}\ \alpha 2 \quad \frac{P\ \{X_i(n+1)=\alpha 2,\ X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}}{P\ \{X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}}$$

$$\times\ P\ \{X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}$$

$$= \Sigma_{\alpha 2}\Sigma_{\alpha 1}\Sigma_{\beta}\ \alpha 2\ P\ \{X_i(n+1)=\alpha 2,\ X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}$$

$$= \Sigma_{\alpha 2}\alpha 2\ \Sigma_{\alpha 1}\Sigma_{\beta}\ P\ \{X_i(n+1)=\alpha 2,\ X_i(n)=\alpha 1,\ X_{i-1}(n)=\beta\}$$

$$= \Sigma_{\alpha 2}\alpha 2\ P\ \{X_i(n+1)=\alpha 2\}$$

$$= E\ [\ X_i(n+1)\ ]$$

Therefore, using the above relation, we can derive the following unconditional relations from equations (4.1.10) through (4.1.12).

i) when $i = 2, 3, \ldots, b, \neq k$

$$E\ [\ E\ [X_i(n+1)\ |\ X_i(n),\ X_{i-1}(n)]\ ]$$

$$= E\ [X_i(n+1)]$$

$$= \left(1 - \frac{i}{n+1}\right) E\ [X_i(n)] + \frac{i-1}{n+1} E\ [X_{i-1}(n)] \qquad (4.1.13)$$

Similarly,

ii) when $i = 1$

$$E\ [X_1(n+1)] = \left(1 - \frac{1}{n+1}\right) E\ [X_1(n)] \qquad (4.1.14)$$

iii) when $i = k$

$$E\ [X_k(n+1)]$$

$$= \left(1 - \frac{k}{n+1}\right) E[X_k(n)] + \frac{k-1}{n+1} E[X_{k-1}(n)]$$

$$+ \frac{2\ b}{n+1} E[X_b(n)] \qquad (4.1.15)$$

The probability, $P_i(n)$, that an insertion falls into one of the key intervals of a CI^i is obtained using the relation $P_i(n) = i\ E\ [X_i(n)]/\ (n+1)$. Using this relation, we can convert the expected difference equations (4.1.13) through (4.1.15) into probability difference equations (4.1.16) through (4.1.18).

$$P_i(n) = \frac{n-i}{n+1} P_i(n-1) + \frac{i}{n+1} P_{i-1}(n-1) \qquad (4.1.16)$$

Similarly, for i = 1 and i = k, the following difference equations are obtained.

$$P_1(n) = \frac{n-1}{n+1} P_1(n-1) \qquad (4.1.17)$$

$$P_k(n) = \frac{n-k}{n+1} P_k(n-1) + \frac{k}{n+1} P_{k-1}(n-1) + \frac{2\ k}{n+1} P_b(n-1) \qquad (4.1.18)$$

Let $\vec{P}(n)$ be a b-component column vector containing $P_i(n)$. Then, equations (4.1.16) through (4.1.18) can be written in a recurrence relation using vectors and matrix.

$$\vec{P}(n) = \left[\ \mathbf{I} + \frac{\mathbf{A}}{n+1}\ \right] \vec{P}(n-1), \qquad (4.1.19)$$

where

$$\vec{P}(n) = [\ P_1(n),\ P_2(n),\ \ldots,\ P_i(n),\ \ldots,\ P_b(n)\ ]^T,$$

I is an b x b identity matrix, and

A is the transition matrix.

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \text{rows} \end{array} \begin{bmatrix} -2 & 0 & & & & & & & \\ 2 & -3 & & & & & & 0 & \\ & 3 & -4 & & & & & & \\ & & \ddots & \ddots & & & & & \\ & & & i & -(i+1) & & & & \\ & & & & \ddots & \ddots & & & \\ & & & & & k & -(k+1) & & 2k \\ & & & & & & \ddots & \ddots & \\ & & 0 & & & & & b & -(b+1) \end{bmatrix} \begin{array}{l} 1 \\ 2 \\ 3 \\ \\ i \\ \\ k \\ \\ b \end{array}$$

The equation (4.1.19) is similar to the equation derived in Eisenbarth et al. (1982) for the analysis of 2-3 trees and in Baeza-Yates (1989a, 1989b) for the analysis of B^+-trees.

Since $P_i(j) = 0$ for all integer j, $(j < i)$, the solution of equation (4.1.19) is obtained by recursion

$$P_i(n) = \prod_{m=0}^{n-i-1} \left[\mathbf{I} + \frac{\mathbf{A}}{n+1-m} \right] \vec{P}(i), \qquad (4.1.20)$$

where $i = 1, 2, \ldots, b$, and

$$\vec{P}(i) = [\, P_1(i), P_2(i), \ldots, P_i(i), 0, \ldots, 0 \,]^T.$$

The probability $P_i(n)$ converges to the solution of a equation $\mathbf{A}\, x_1 = 0$ when n goes infinite, where x_1 is an

eigenvector corresponding to the eigenvalue $\lambda_1 = 0$. The convergence of $P_i(n)$ is proved in Theorem 4.1.1 (Eisenbarth et al., 1982, p135).

Theorem 4.1.1 Let **A** be the b × b transition matrix of a fringe analysis problem. Let $\lambda_1, \ldots, \lambda_b$ be eigenvalues of **A**, with $\lambda_1=0 \geq Re(\lambda_2) \geq \ldots \geq Re(\lambda_b)$, and let x_1 be the eigenvector of **A** corresponding to λ_1. Then, there exists a constant c such that for every vector P(n)

$$\left| \vec{P}(n) - c\, x_1 \right| = O\left(n^{Re(\lambda_n)} \right),$$

where $\vec{P}(n)$ is defined in equation (4.1.19).

The probability of a CI being split after n^{th} insertion is denoted by $P_b(n)$, and can be derived from equation (4.1.20).

$$P_b(n) = \prod_{m=0}^{n-b-1} \left[\mathbf{I} + \frac{\mathbf{A}}{n+1-m} \right] \vec{P}(b) \qquad (4.1.21)$$

If we know the probability $P_i(n)$, then we can get the expected number of CI's for large n, $E[X_i(n)]$, using the relation $E[X_i(n)] = ((n+1)/i)\, P_i(n)$. The $E[X_i(n)]$ is given by

$$E[X_i(n)] = \frac{n+1}{i} P_i(n) = \frac{n+1}{i} \prod_{m=0}^{n-i-1} \left[\mathbf{I} + \frac{\mathbf{A}}{n+1-m} \right] \vec{P}(i). \qquad (4.1.22)$$

The total expected number of CIs, X(n), which is the sum of $E[X_i(n)]$, is given by

$$X(n) = \sum_{i=1}^{b} E[X_i(n)] = \sum_{i=1}^{b} \frac{n+1}{i} \prod_{m=0}^{n-i-1} \left[I + \frac{A}{n+1-m} \right] \vec{P}(i). \tag{4.1.23}$$

The utility of a file is estimated by the minimum number of CIs and total number of CIs, X(n), in the following way:

$$\text{Utility} = \frac{\text{min. number of CIs}}{X(n)} = \frac{n/b}{X(n)} \tag{4.1.24}$$

For a given number of records, utilization of a file only depends on the CI capacity (b) and the total number of CIs (X(n)).

4.2. Modeling of a Physical Database When Both Insertions and Deletions Are Considered

The modeling described in this section considers both insertions and deletions of records in the construction of a physical database. Different from the analysis in section 4.1, this section adopts continuous transaction analysis. The assumptions used in section 4.1 are also effective in this section. However, the probability of a record being deleted is assumed to be equal for all records. It is also assumed that the arrival of records to be inserted to the

physical database follows a Poisson process with arrival rate λ. Each record in the database and in the arriving stream has a unique key. All the discussions in section 4.1 relating to a record insertion into a CI also apply to this section and are therefore not be repeated. When a record is inserted into a database, its service is assumed to have commenced. The service continues until a record is physically deleted from the system, and the space occupied by the record is then immediately reusable by another record. It is assumed that the service times of all records inserted into the file are independently, identically distributed random variables with an exponential distribution and a finite mean $1/\mu$. The rate that an insertion occurs between time t and $t+\Delta t$ is equal to $\lambda\Delta t$, and the rate that a deletion occurs between this time interval is equal to $\mu\Delta t$.

A major source of difficulty in modeling arises from the need to keep track of the number of CI's. This number is influenced by the rate of arrival of new records into each CI, by the deletion rate of records, and by the way of handling overflows; i.e., CI splits. These in turn depend on the probabilities of a record insertion and deletion into and from each CI. To develop a model, fringe analysis is also used, as in section 4.1. However, the fringe in this section is described by the expected number of CI's at time

t. The birth and death process has been adopted as a means of describing the transitional behavior of fringes.

Let $X_i(t)$ be the number of CI^is. It is assumed that the probability that a record is inserted into a CI^i is $iX_i(t)/(n+1)$, and the probability that a record is deleted from a CI^i is $iX_i(t)/n$ at time t, where n is the total number of records in the system at time t and $n = n(t)$. There are two occasions where $X_i(t)$ can increase. One occurrence is an insertion into the CI^{i-1}s, and the other is a deletion from CI^{i+1}s. Xi(t) decreases when an insertion or a deletion occurs from the CI^is. In other cases, $X_i(t)$ remains the same. This transaction follows a birth and death process. The instantaneous changes in the system state can amount to an increase (birth) or a decrease (death) of a CI^i. When a transaction occurs, the following conditional transition probabilities are given using a state of $X_i(t+\Delta t)$ for a given state $X_i(t)$, where $i = 2,3, \ldots, b-1$ and b is the maximum number of records that a CI can hold, as defined in section 4.1.

$$\Pr \{ X_i(t+\Delta t) = a2 \mid X_i(t) = a1,\ X_{i-1}(t),\ X_{i+1}(t) \}$$

$$= \left[\frac{(i-1)X_{i-1}(t)}{n+1} \lambda_n \Delta t \right] + \left[\frac{(i+1)X_{i+1}(t)}{n} \mu_n \Delta t \right] \qquad (4.2.1)$$

$$\text{for } a2 = a1 + 1$$

$$= \left[\frac{iX_i(t)}{n+1} \lambda_n \Delta t \right] + \left[\frac{iX_i(t)}{n} \mu_n \Delta t \right] \qquad (4.2.2)$$

for a2 = a1 - 1

$$= 1 - (4.2.1) - (4.2.2), \qquad (4.2.3)$$

for a2 = a1

where λ_n and μ_n are the insertion rate and the deletion rate, respectively, and a1 and a2 are constant.

The conditional probability equations (4.2.1) through (4.2.3) can be interpreted in the following ways. The first term of the equation (4.2.1) represents the birth of a record into one of the CI^{i-1}s, with a probability $(i-1)X_{i-1}(t)\ \lambda_n\Delta t / (n+1)$. The second term represents the death at one of the CI^{i+1}s, with a probability $(i+1)X_{i+1}(t)\ \mu_n\Delta t / n$. The result of these combinations is an increase in the number of CI^is. The interpretation of equation (4.2.2) is similar to that of equation (4.2.1). This equation represents the birth or the death of a record at one of the CI^is with a probability $iX_i(t)\ \lambda_n\Delta t / (n+1)$ or $iX_i(t)\ \mu_n\Delta t / n$, respectively. The result is a decrease in the number of CI^is. In other cases that do not appear in equation (4.2.1) and (4.2.2), the number of CI^is remains the same.

When i = 1, i = b, or i =k, the equation (4.2.1) is modified in the following way.

i) When i = 1

$$\Pr \{ X_1(t+\Delta t) = a2 \mid X_1(t) = a1,\ X_2(t) \}$$

$$= \left[\frac{2\ X_2(t)}{n} \mu_n \Delta t \right] \qquad (4.2.4)$$

The interpretation of equation (4.2.4) is that the increase of $X_1(t)$ is caused only by the deletion on one of the CI^2s with a probability $2X_2(t)\ \mu_n\Delta t\ /\ n$.

ii) When i = b

$$\Pr \{ X_b(t+\Delta t) = a2 \mid X_b(t) = a1,\ X_{b-1}(t) \}$$

$$= \left[\frac{(b-1)X_{b-1}(t)}{n+1} \lambda_n \Delta t \right] \qquad (4.2.5)$$

The interpretation of equation (4.2.5) is that the increase of $X_b(t)$ is caused only by the insertion on one of the CI^{b-1}s with a probability $(b-1)X_{b-1}(t)\ \lambda_n\Delta t\ /\ (n+1)$.

iii) When i = k, one more equation is added to equations (4.2.1), (4.2.2), and (4.2.3) to reflect the CI split.

$$\Pr \{ X_i(t+\Delta t) = a2 \mid X_i(t) = a1,\ X_{i-1}(t),\ X_{i+1}(t),\ X_b(t) \}$$

$$= \left[\frac{b\ X_b(t)}{n+1} \lambda_n \Delta t \right] \qquad (4.2.6)$$

$$\text{for } a2 = a1 + 2$$

The interpretation of equation (4.2.6) is that the increase of $X_k(t)$ has an additional source besides the sources in equation (4.2.1). The birth of a record into one of the CI^bs, with a probability $bX_{b-1}(t)\ \lambda_n\Delta t\ /\ (n+1)$, causes a CI split and, with this split, $X_k(t)$ is increased.

Using the conditional probabilities (4.2.1) through (4.2.6), the expected values of $X_i(t+\Delta t)$, conditional on $X_i(t)$, $X_{i-1}(t)$ and $X_{i+1}(t)$, can be obtained as

$$
\begin{aligned}
&E\left[\, X_i(t+\Delta t) \mid X_{i-1}(t),\, X_i(t),\, X_{i+1}(t) \,\right] \\
&= (X_i(t)+1)\left\{\left[\frac{(i-1)X_{i-1}(t)}{n+1}\lambda_n\Delta t\,\right] + \left[\frac{(i+1)X_{i+1}(t)}{n}\mu_n\Delta t\,\right]\right\} \\
&\quad + (X_i(t)-1)\left\{\left[\frac{iX_i(t)}{n+1}\lambda_n\Delta t\,\right]\left[\frac{iX_i(t)}{n}\mu_n\Delta t\,\right]\right\} \\
&\quad + X_i(t)\,\{\,1 - (4.2.1) - (4.2.2)\,\} \\
&= \frac{(i-1)X_{i-1}(t)}{n+1}\lambda_n\Delta t + \frac{(i+1)X_{i+1}(t)}{n}\mu_n\Delta t - \frac{iX_i(t)}{n+1}\lambda_n\Delta t \\
&\quad - \frac{iX_i(t)}{n}\mu_n\Delta t + X_i(t), \qquad (4.2.7)
\end{aligned}
$$

where $i = 2, 3, \ldots, b-1, \neq k$.

Using the expectation from equation (4.2.7), we can obtain the unconditional expectation for $X_i(t+\Delta t)$. That is, $E\,[E\,[\,X_i(t+\Delta t) \mid X_{i-1}(t),\, X_i(t),\, X_{i+1}(t)\,]]$ becomes $E\,[X_i(t+\Delta t)]$. (See section 4.1 for proof of this.) Therefore, the expectation on equation (4.2.7) can be represented by the following equation:

$$
\begin{aligned}
&E\left[\, E\left[\, X_i(t+\Delta t) \mid X_{i-1}(t),\, X_i(t),\, X_{i+1}(t)\,\right]\,\right] \\
&\quad = E\left[\, X_i(t+\Delta t)\,\right] \\
&\quad = \frac{(i-1)\lambda_n\Delta t}{n+1}E\left[X_{i-1}(t)\right] + \frac{(i+1)\mu_n\Delta t}{n}E\left[X_{i+1}(t)\right]
\end{aligned}
$$

$$- \frac{i\ \lambda_n \Delta t}{n+1} E\ [X_i(t)] - \frac{i\ \mu_n \Delta t}{n} E\ [X_i(t)]$$

$$+ E\ [X_i(t)] \qquad (4.2.8)$$

If we subtract E [X(t)] from both sides of equation (4.2.8) and divide them by Δt, we will have

$$\frac{E\ [X_i(t+\Delta t)] - E\ [X_i(t)]}{\Delta t}$$

$$= \frac{(i-1)\lambda_n}{n+1} E\ [X_{i-1}(t)] + \frac{(i+1)\mu_n}{n} E\ [X_{i+1}(t)]$$

$$- \frac{i\ \lambda_n}{n+1} E\ [X_i(t)] - \frac{i\ \mu_n}{n} E\ [X_i(t)]. \qquad (4.2.9)$$

If we take the limit on Δt, we will have the following differential difference equation:

$$\lim_{\Delta t \to 0} \frac{E\ [X_i(t+\Delta t)] - E\ [X_i(t)]}{\Delta t} = \frac{dE\ [X_i(t)]}{dt}$$

$$= \frac{(i-1)\ \lambda_n}{n+1} E\ [X_{i-1}(t)] + \frac{(i+1)\ \mu_n}{n} E\ [X_{i+1}(t)]$$

$$- \left(\frac{i\ \lambda_n}{n+1} + \frac{i\ \mu_n}{n} \right) E\ [X_i(t)] \qquad (4.2.10)$$

Substitute E $[X_i(t)]$ with $Y_i(t)$ and let λ_n and μ_n be λ and μn. Then, equation (4.2.10) can be rewritten as

$$Y_i'(t) = \alpha\ (i-1)\ Y_{i-1}(t) + \mu\ (i+1)\ Y_{i+1}(t) - (\alpha+\mu)\ i\ Y_i(t), \qquad (4.2.11)$$

where $Y_i'(t) = dY_i(t)/dt$, $\alpha = \lambda\ /\ (n+1)$, and $i = 2,3,\ldots,b-1, \neq k$. This equation is similar to the equation that Quitzow and Klopprogge (1980) derived in terms of pages considering both insertions and deletions for the B-trees model.

When $i = 1$ or $i = b$, we can derive equation (4.2.12) or (4.2.13) using a procedure similar to that used in equation (4.2.7) through (4.2.10) to derive equation (4.2.11).

$$Y_1'(t) = 2\ \mu\ Y_2(t) - (\alpha + \mu)\ Y_1(t) \qquad (4.2.12)$$

$$Y_b'(t) = \alpha\ (b-1)\ Y_{b-1}(t) - (\alpha + \mu)\ b\ Y_b(t) \qquad (4.2.13)$$

If an insertion occurs on one of the CI^bs, then a CI^b splits. When this happens, half the records remain in the split CI, and the other half move to a new CI within the same CA or to that in a different CA, depending on space availability as explained earlier in Chapter 2. As a result of the split of one of the CI^bs, two CI^ks are generated. Therefore, when $i = k$, this CI split effect is added to the differential equation in the following way:

$$Y_k'(t) = \alpha\ (k-1)\ Y_{k-1}(t) + \mu\ (k+1)\ Y_{k+1}(t) - (\alpha + \mu)\ k\ Y_k(t) + 2\ \alpha\ b\ Y_b(t). \qquad (4.2.14)$$

The graphical representation of the changes of $Y_i(t)$ in Δt is depicted in Figure 4.2.1. This figure can be interpreted in the following way:

i) Circles with Y_i represent the states of $Y_i(t)$, $(1 \le i \le b)$.

ii) A labeled arc from Y_{i-1} to Y_i illustrates the increase of Y_i in Δt and a labeled arc from Y_i to Y_{i-1} illustrates the decrease of Y_i in Δt, $(1 \le i \le b)$.

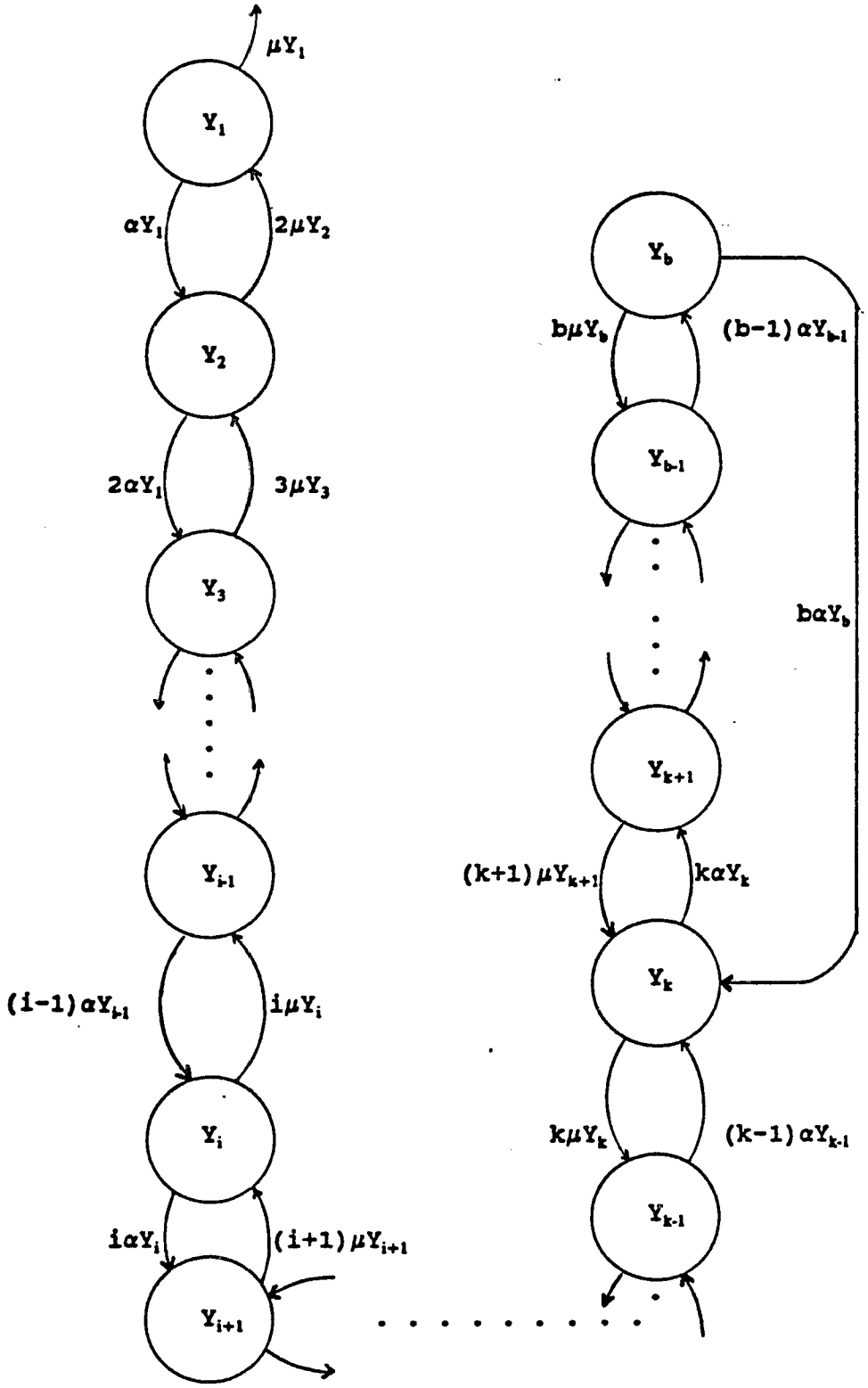
iii) A labeled arc from Y_1 to nowhere indicates the release of a CI^1 to the future available space.

iv) A labeled arc from Y_b to Y_k illustrates the increase of Y_k in Δt as a result of a CI split.

The solution for the system of differential difference equations given in equations (4.2.11) through (4.2.14) can be obtained by an initial condition method. Consider a database that was initially created with n_0 fixed-length records. Let F(0) denote the total number of CIs assigned when ξ number of records are initially loaded into a CI, where ξ is called the loading factor. Since each CI has ξ records on file creation, it allows that F(0) equals to $\lceil n_0/\xi \rceil$. The initial conditions (i.c.) that reflect this fact can be set up by

$$\begin{aligned} Y_i(0) &= F(0) && \text{for } i = \xi \\ Y_i(0) &= 0 && \text{otherwise.} \end{aligned} \qquad (4.2.15)$$

Figure 4.2.1 Graph for the basic model.



Equations (4.2.11) through (4.2.14) can be represented as a normalized system of b first-order, linear equations. These equations form an initial condition problem with given initial conditions in equation (4.2.15).

This initial condition problem can be represented in a vector form:

$$\vec{Y}'(t) = \vec{A}(t)\ Y(t), \qquad (4.2.16)$$

i.c.

$$\vec{Y}(0) = [0\ 0\ \ldots\ F(0)\ \ldots\ 0\ 0]^T,$$

where

$$\vec{Y}(t) = [Y_1(t)\ Y_2(t)\ \ldots\ Y_b(t)]^T$$

and

$$\mathbf{A}(t) = \begin{bmatrix} -\beta & 2\mu & & & & 0 & \\ \alpha & -2\beta & 3\mu & & & & \\ & \ddots & & & & & \\ & & (i-1)\alpha & -i\beta & (i+1)\mu & & \\ & & & \ddots & & & \\ & & & & (k-1)\alpha & -k\beta & (k+1)\mu & 2\alpha b \\ & & & & & \ddots & & \\ 0 & & & & & & \ddots & \\ & & & & & & (b-1)\alpha & -b\beta \end{bmatrix} \begin{matrix} \text{rows} \\ 1 \\ 2 \\ \\ i \\ \\ k \\ \\ \\ b \end{matrix}$$

where $\alpha = \lambda/(n+1)$, $\beta = \alpha + \mu$, and $n = n(t)$.

The number of records in the database at time t can be obtained by solving the following ordinary differential equation.

$$\frac{dn}{dt} = \lambda - \mu n \qquad (4.2.17)$$

Equation (4.2.17) indicates that insertion increases, and deletion decreases, the number of records in the system. The general solution of this equation is given by

$$n = \lambda/\mu + C \exp(-\mu t), \qquad (4.2.18)$$

where C is an arbitrary constant that should be determined by the initial condition. As the database was created with n_0 records initially, we can decide C using the initial condition; i.e., when t=0, $C = n_0 - \lambda/\mu$. Using this constant C, we can determine the particular solution of equation (4.2.17).

$$n = \lambda/\mu + (n_0 - \lambda/\mu) \exp(-\mu t) \qquad (4.2.19)$$

The system of differential equation (4.2.16) could not be solved in an analytical method, because n is a function of time and so is the **A** matrix. Numerical method is used to obtain the solution of this equation, and the simulation package SLAM II has been used for this purpose. SLAM II uses Runge-Kutta-Fehlberg (RKF) algorithms to integrate the equations (Pritsker, 1986). The RKF method is particularly useful if certain coefficients in the differential equation are empirical functions for which analytical expressions are not known, and hence for which initial series developments

are not feasible. This method provides the specific capability of numerically integrating a system of first-order ordinary differential equations such as equation (4.2.16). Using this method, a simulation is done to obtain the time history of Y(t) for a given equation for Y'(t). This is accomplished by considering Y(t) as a function of the derivatives of Y(t) using a Taylor series expansion. The values of Y(t) can be estimated by evaluating the Y'(t). Given the vector of expected number of CIs, Y(t), the total number of CIs could be calculated. Let F(t) represent the total number of CIs at a given time. Then, F(t) is given by

$$F(t) = \sum_{i=1}^{b} Y_i(t), \qquad (4.2.20)$$

and the minimum number of CIs, CI_{min}, is given by

$$CI_{min} = \sum_{i=1}^{b} i\, Y_i(t) \;/\; b. \qquad (4.2.21)$$

From equations (4.2.20) and (4.2.21), utility of the file could be calculated. Utility represents that portion of the file that is utilized or being used efficiently.

$$\text{Utility} = \frac{CI_{min}}{F(t)} = \sum_{i=1}^{b} i\, Y_i(t) \;/\; \left(b \sum_{i=1}^{b} Y_i(t) \right) \qquad (4.2.22)$$

A detailed analysis to the results given so far is presented in Chapter 5 through numerical experiments.

Chapter 5

NUMERICAL EXPERIMENT RESULTS FOR THE MODELS OF PHYSICAL DATABASE DESIGN

The previous chapter focusses on developing models that describe the behavior of a physical database in a VSAM file. One model considers the case where the database is built up through the insertions of records without deletions. The other model considers the case where the database is built up through both insertions and deletions of records. In developing models, two factors are vital. One is the probability of inserting a record into a particular CI, and the other is the CI split. The increase and the decrease of the number of records for each CI^is depend mainly on the probabilities of inserting and deleting records. The CI splits adopted as an alternative technique of handling the overflow problem act as another source of increment for the number of CI^ks.

An efficient design for a physical database system is a function of the various parameters that define the system. A specific physical database can be described by the values that are assigned to the various parameters. Design parameters define the particular implementation that has been chosen for a given application. These design parameters include the capacity of a CI, loading factors, the method of handling the overflows, etc.

In this chapter, we introduce details of computational results for each model. The primary objective of these numerical analysis is the verification and the understanding of the models developed in Chapter 4 by showing how the physical database behaves with changes of different parameter values. The examination is based only on the models developed in Chapter 4, however; we neglect the status changes related to the CA splits in this chapter and discuss them in the following chapters. The first model was verified using a program written in Pascal, and the second model was verified using the simulation package SLAM II.

5.1. Numerical Analysis for the Physical Database Constructed with Records Insertions Only

The insertion of a record into the database contributes to an increase in the number of CIs in the system in the following ways. As explained in Section 4.1, the insertion of a record into the CI^1s contributes to an increase in the number of CI^2s and to decrease in the number of CI^1s. Similarly, the insertion of a record into the CI^2s increases the number of CI^3s and decreases the number of CI^2s and so on. However, the situation is somewhat different when a record is inserted into the CI^bs. In this case, because of the CI split, the number of CI^ks increases and the number of CI^bs decreases. From the above mechanism, we can conjecture

the following results. With a continuous increase in record insertions into the system, the number of CI's, $X_i(n)$ ($1 \le i < k$) decrease to zero because there are no resources that contribute to the increase of them. However, $X_i(n)$ ($k \le i \le b$) continuously increases as a result of records supply and CI splits. This conjecture is well matched to the numerical analysis results, which are based on the given model, and is verified in Table 5.1.1. This table was developed using the model in Section 4.1. Eleven was used for the capacity of a CI; i.e., b=11. When $n \le 400$, $X_i(n)$ ($1 \le i \le 5$) retains ≥ 0.005, although they are negligibly small. When $n \ge 500$, however, they become practically zero, whereas $X_i(n)$ ($6 \le i \le 11$) grow bigger with the increase of n. This situation is depicted in Figure 5.1.1. As this figure indicates, the total number of CIs in the system is shown in Figure 5.1.2. The interesting finding from this figure is that the total number of CIs increases linearly with the increase of the total number of records in the system.

Table 5.1.1 also shows the probability values $P_i(n)$ for the various number of records in the system, where i=1,2, ...11. As the number of records in the system increases, $P_i(n)$ ($1 \le i \le 5$) decrease and become zero when $n \ge 200$ while $P_i(n)$ ($6 \le i \le 11$) have stable values convergent to 0.219, 0.191, 0.170, 0.153, 0.139, and 0.127. Figure 5.1.3 depicts the distribution of these probabilities. Practically, this situation indicates that, when n becomes ≥ 200, all the CIs

contain more than 5 records; thus, we no longer need to consider $X_i(n)$ ($1 \le i \le 5$).

The storage utilization, which was calculated using the equation (4.1.24), is shown in Table 5.1.1 and in Figure 5.1.4. The storage utilization increases as n increases. However, when n is large enough, the utility becomes stable, approaching a value of around 0.71. This analysis can be comparable with B-tree analysis when n is large. As $X_i(n)$ ($1 \le i \le 5$) become zero with large n, it is plausible that the physical database behaves similarly to B-tree. Then, we can compare the utility values from the model with those from the B-tree analysis. The predicted utility values used as a standard for B-tree analysis are Ln2 + O(1/b) (Baeza-Yates 1989a, 1989b; Chu and Knott 1989; Eizenbarth et al. 1982). The values we have shown above are within the error range of this prediction value. This supports the fact that the model in Section 4.1. is reasonably formulated. Another numerical experiment related to the utility analysis was done for the files that have different CI capacities. Table 5.1.2 shows the different utility values and the total number of CIs for each file when the total number of records in the system is fixed at 200. This table shows the negative relation between utility and the size of b. With a smaller value of b, we have a higher utility value; i.e., when b equals 21, utility is 0.6960, but when b equals 3, utility is 0.7739, and so on. However, if we adopt a small

CI capacity, the probability of a CI split increases, as does the total number of CIs. The result will be an increase in the record insertion cost because of the high CI splits. Consequently, the optimal choice of the CI capacity will be the tradeoff between the high utility and the high record storage cost.

Table 5.1.1 Probability of Inserting a record into CI's, the number of CI's, and associated storage utilization

(N)	20	50	100	200	300	400	500
P1(n)	0.0048	0.0008	0.0002	0.0000	0.0000	0.0000	0.0000
P2(n)	0.0095	0.0016	0.0004	0.0001	0.0000	0.0000	0.0000
P3(n)	0.0143	0.0024	0.0006	0.0002	0.0001	0.0000	0.0000
P4(n)	0.019	0.0031	0.0008	0.0002	0.0001	0.0000	0.0000
P5(n)	0.0238	0.0039	0.001	0.0002	0.0001	0.0001	0.0000
P6(n)	0.186	0.2174	0.2183	0.2186	0.2187	0.2187	0.2187
P7(n)	0.1735	0.1891	0.1909	0.1913	0.1913	0.1913	0.1913
P8(n)	0.1695	0.1677	0.1696	0.17	0.17	0.1701	0.1701
P9(n)	0.1569	0.151	0.1526	0.153	0.153	0.1531	0.1531
P10(n)	0.1344	0.1373	0.1386	0.139	0.1391	0.1391	0.1391
P11(n)	0.1082	0.1256	0.127	0.1274	0.1275	0.1275	0.1275
X1(n)	0.1	0.04	0.02	0.01	0.0067	0.005	0.004
X2(n)	0.1	0.04	0.02	0.01	0.0067	0.005	0.004
X3(n)	0.1	0.04	0.02	0.01	0.0067	0.005	0.004
X4(n)	0.1	0.04	0.02	0.01	0.0067	0.005	0.004
X5(n)	0.1	0.04	0.02	0.01	0.0067	0.005	0.004
X6(n)	0.651	1.848	3.6753	7.3235	10.9695	14.615	18.2603
X7(n)	0.5205	1.378	2.7547	5.4919	8.2266	10.9609	13.6949
X8(n)	0.4451	1.0693	2.1415	4.2708	6.398	8.5248	10.6513
X9(n)	0.3662	0.8558	1.7123	3.4161	5.118	6.8195	8.5208
X10(n)	0.2822	0.7003	1.4001	2.7945	4.1871	5.5794	6.9714
X11(n)	0.2066	0.5824	1.1657	2.3282	3.489	4.6492	5.8093
Total	2.9716	6.6338	12.9496	25.675	38.4215	51.1738	63.928
Utility	0.6119	0.6852	0.702	0.7082	0.7098	0.7106	0.7113

Table 5.1.2 Utility changes for each CI capacity size

Capacity	Utility	Prob. of split	Total no. of CIs
3	0.7739	0.4285	86
5	0.7361	0.2702	54
7	0.7212	0.1969	40
9	0.7132	0.1548	31
11	0.7082	0.1274	26
13	0.7046	0.1082	22
15	0.7018	0.094	19
17	0.6995	0.0831	17
19	0.6976	0.0746	15
21	0.696	0.0678	13

Figure 5.1.1 Number of CI's

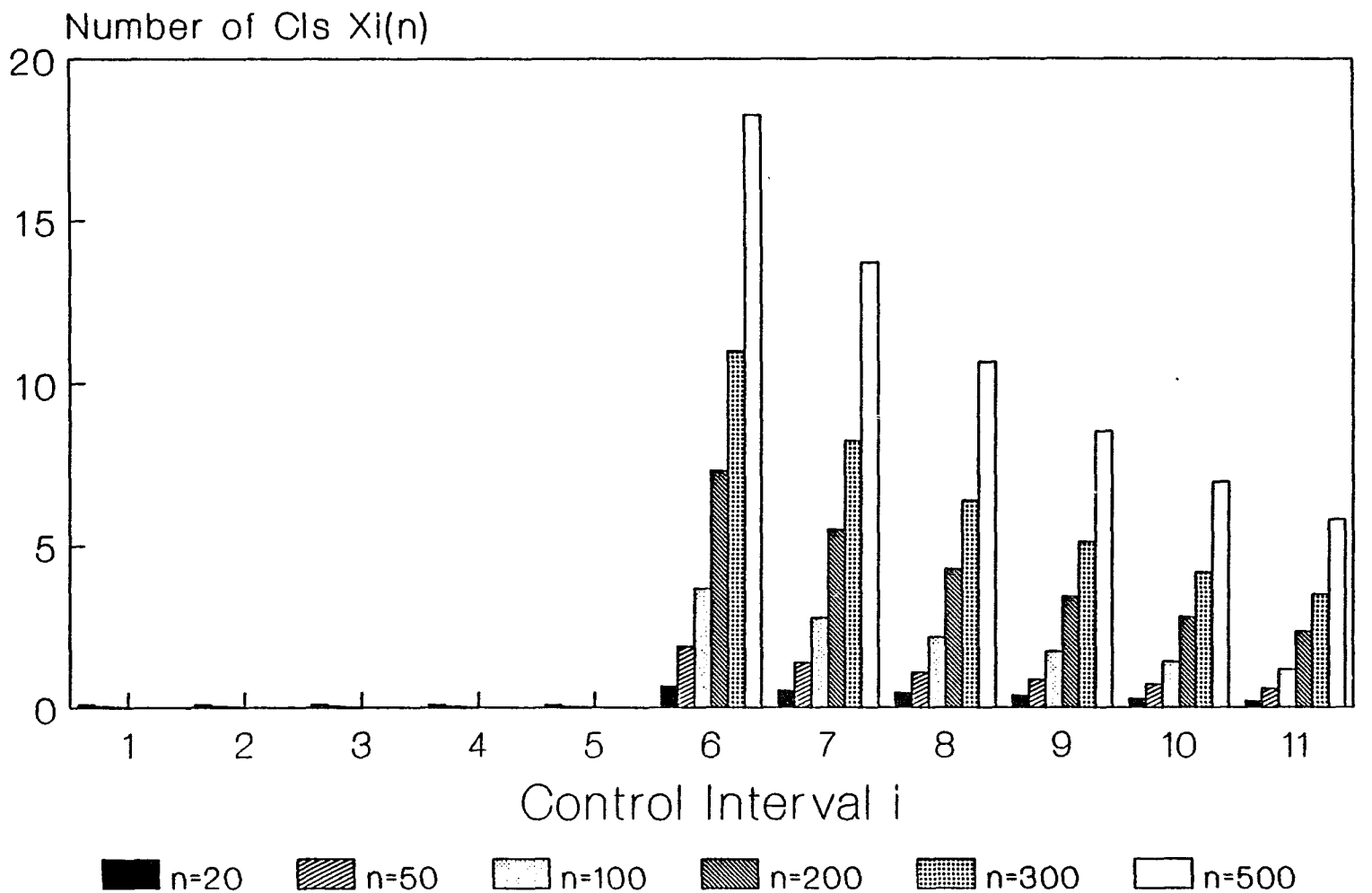


Figure 5.1.2 The relation between the number of total records and the total number of CIs

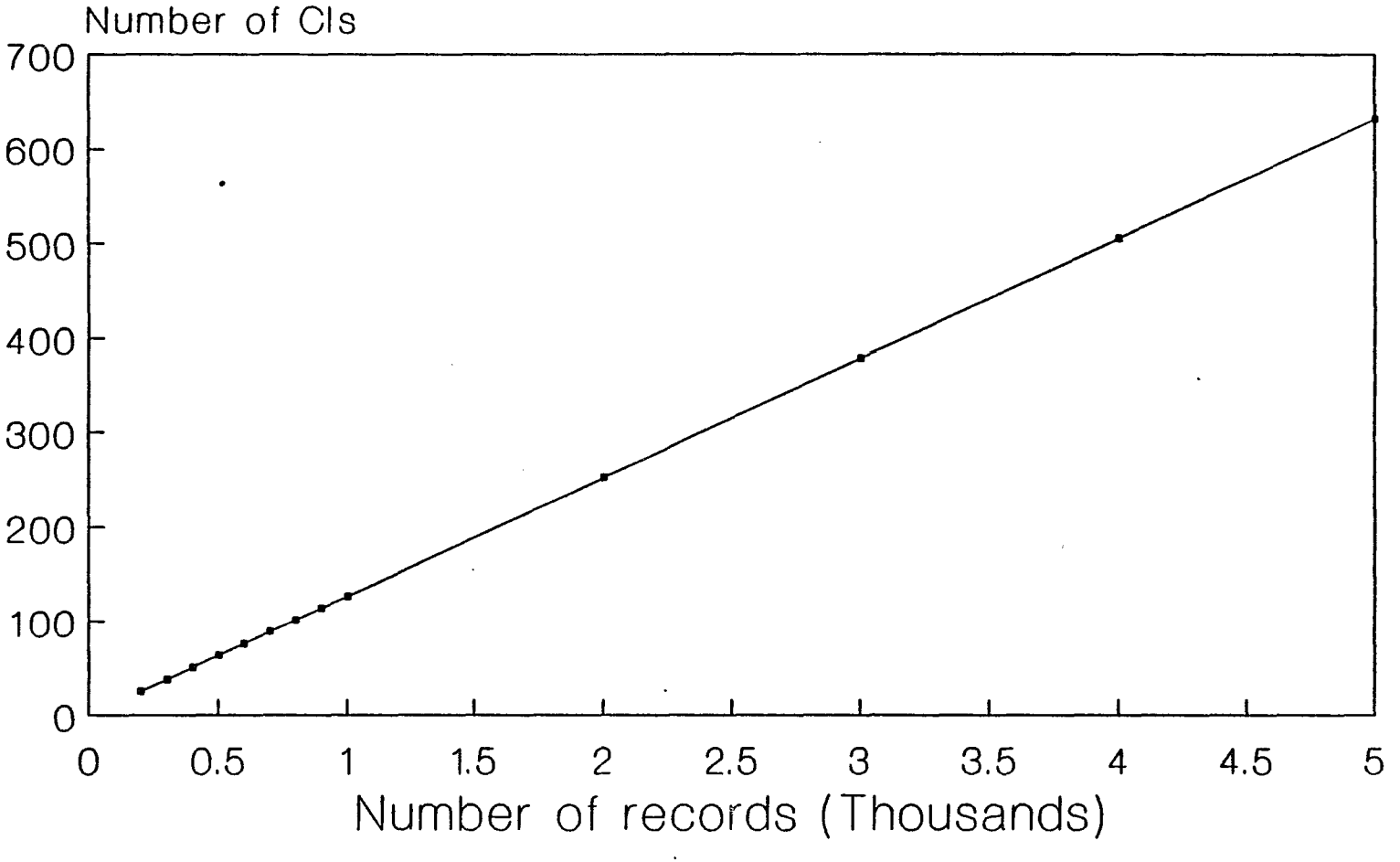


Figure 5.1.3 Probabilities of inserting a record into a CI's

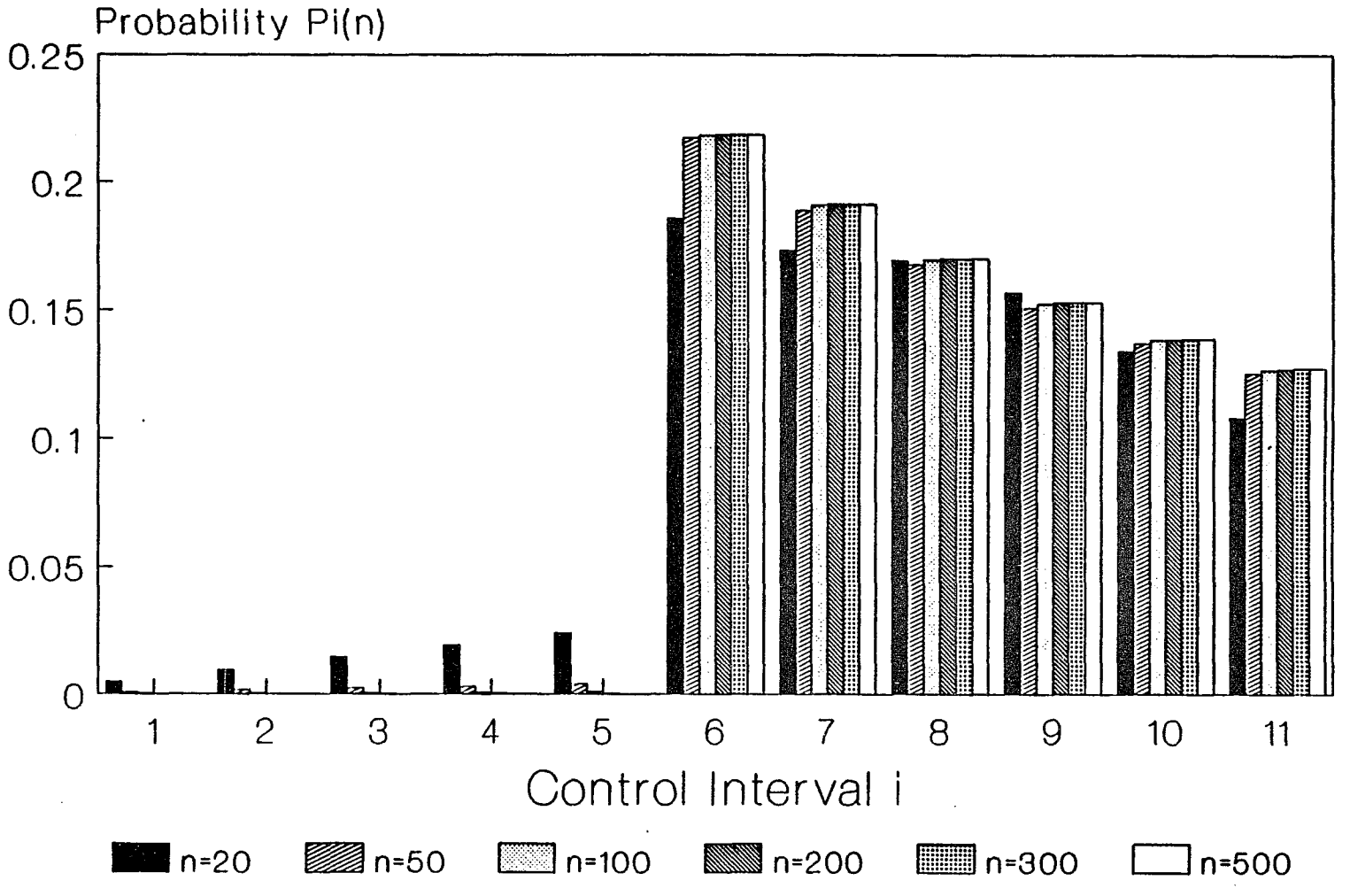
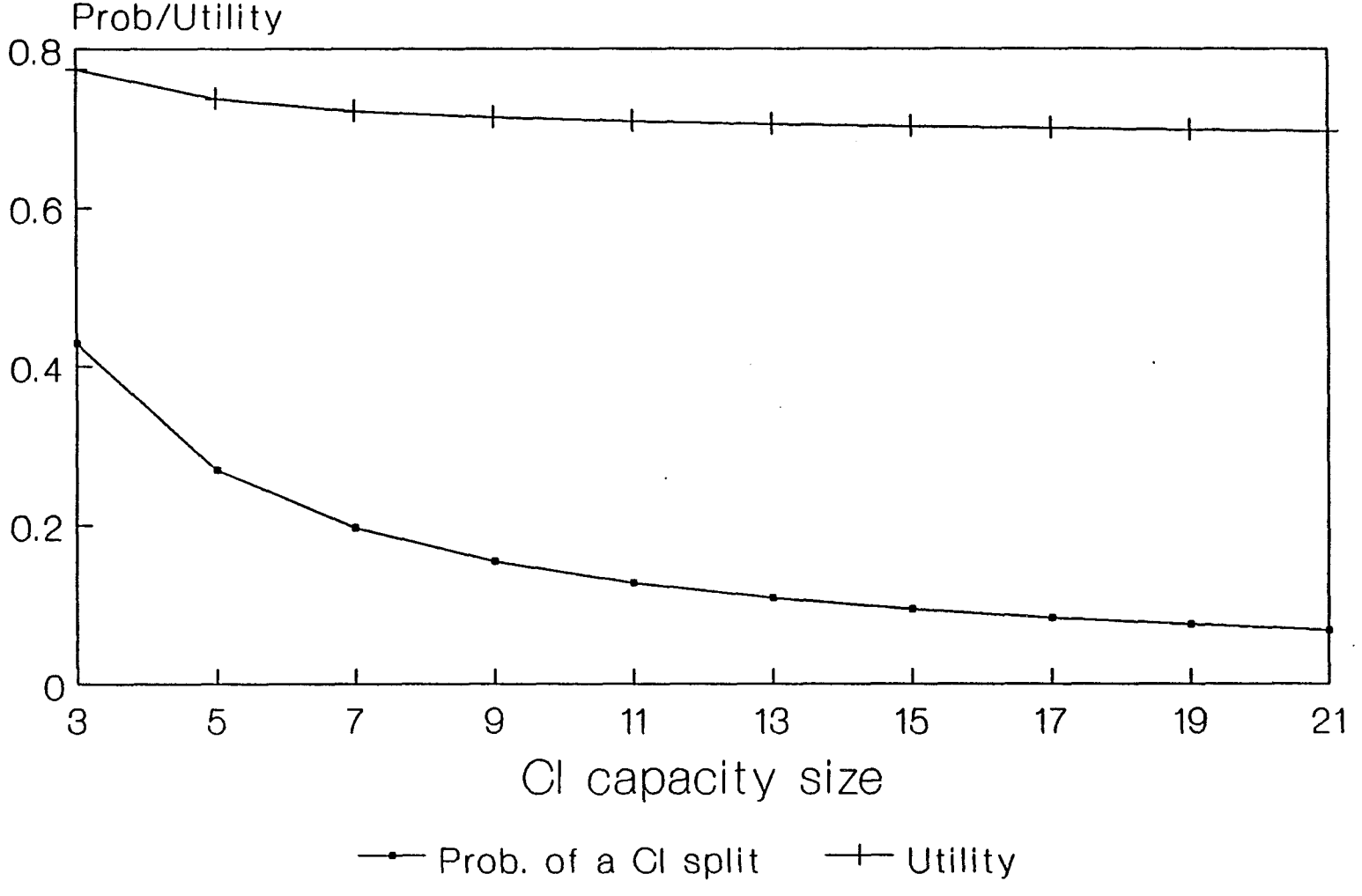


Figure 5.1.4 Utility changes and the probability of a CI split for different CI capacity size.



5.2. Numerical Analysis for the Physical Database Constructed with Records Insertions and Deletions

Consider a database that has 50,000 records initially. This size of this database file is small to medium by conventional commercial standards. The two key parameters that determine the evolution of the physical database system from state to state are the record arrival rate to the database and the deletion rate of records from the database. In the development of the model in Section 4.2, the arrival rate per period has been assumed to be distributed as a Poisson random variable, and the deletion time distribution per record has been assumed to be exponentially distributed. For the numerical experiments, it is assumed that the arrival rate of records to be inserted into the database is 200 records per period. The unit of a period is assumed to be one hour. Each record is assumed to be deleted at a rate of 0.001 per period. This means that if there are N_0 records in the system, the rate at which records are deleted is $0.001\ N_0$, and this rate is changed as the number of records in the system is changed over time. The choice of the values of the insertion rate and deletion rate combination is based on the assumption that the database grows over the time horizon. Besides the two parameters mentioned above, we need to consider two other parameters: CI capacity sizes, and the range of loading factors for a given CI capacity

size. In performing the numerical analysis for the model given in Section 4.2, a finite set of parameter values is considered to be tested, as was done in Section 5.1. Verification of the model could be done using the simulation package SLAM II. To verify the model, we can compare the total number of records that can be predicted using the equation (4.2.19) with the total number of records that is calculated from the simulation. From the simulation, we can obtain the number of CI^is. Summing up i times CI^is for all i's gives the total number of records.

The growth of a database over time from a given initial state is illustrated in Figure 5.2.1 and in Table 5.2.1 for different CI capacity values when λ=200, μ=0.001, and N_0=50,000. To find the effect of the CI capacity sizes, we experimented with three different sizes: 9, 15, and 21. To set up the same initial condition in the analysis, we arranged the number of records to be loaded initially as 6, 12, and 14, respectively, yielding utility of 0.6667. Under this initial condition with 50,000 records, the total initial number of CIs for each database is 8,334, 5,000, and 3,572, respectively. As can be seen in Figure 5.2.1, the number of CIs increases as time increases for all three cases. The number of CIs in the file with capacity 9 increases faster than the others. However, the relative increases in the total number of CIs, compared to the initial number of CIs, are similar for all files. Let us

define state variable $s_t \equiv X_t/X_0$, where X_0 is the number of CIs at the initial file loading time and X_t is the total number of CIs at time t. Then, from the Table 5.2.1, we can obtain $s_{200} = 1.5$, $s_{300} = 1.8$, and $s_{400} = 2.0$ for all files. The file size is doubled at time 400 hours regardless of the CI sizes. This fact gives us the possibility of predicting the number of CIs in a database at the desired time if we are given the state variable s_t. Another comparison of the behaviors of files with different CI capacities is shown in Table 5.2.2 and in Figure 5.2.2, which depict the utility changes over the time horizon for each file. As indicated, the time horizon can be divided into three parts according to the relative pattern of utilities of the files. In the early stage of the file life, utilities increase continuously until they reach their maximum utility values at a time between 60 hours and 70 hours, and decrease thereafter. The greater the CI capacity, the higher the utility values, until the time about 170 hours. At this point, utility values for all files become almost the same with a value of 0.69. After this time, the utility patterns change. Utility is higher for the file that has a smaller CI capacity until the time 300 hours. After the time 300 hours, the file with the greater CI capacity has higher utility values again. What we determined from these observations is as follows. First, the utilities are greater than the initial utility until the time of 300

hours. Thereafter, they decline to a value less than that of the initial utility; i.e., file utilization deteriorates after the time of 300 hours. Second, if the utility is a measure of the efficient file use related to the CI size, the optimal CI size will be a function of time. Although the choice of favorable CI size is a function of time, if the length of the time period and the magnitude of the utility differences among the different files of different CI sizes are considered, one would recommend use of the large CI size. This conclusion can be supported also by a comparison of the total number of CIs in each file as follow. The increase in the total number of CIs during the 500 hours period for the file with b=9 is 10,526 and for the file with b=21 is 4,259. This indicates that the file with b=9 experienced more costly CI splits than the file with b=21.

To see how the file behaves for the different loading factors, we experimented with initial loading of 9, 10, 11, 12, 13, and 14 records for the file whose CI capacity is 15. It seems evident that, if we use a smaller loading factor, we need a greater number of CIs. However, this is true only for the file that has a short life time. If the file life is more than 400 hours, the total number of CIs becomes almost the same regardless of the loading factor (Table 5.2.3 and Figure 5.2.3). A similar situation could be obtained for the utility analysis. At the initial stage of

the file, it is difficult to predict the relationship between the utility and the loading factor. However, when the time period becomes long enough, the utilities of all cases converge to one value. The utilities become about 0.67 at time 300 hours and about 0.65 at time 500 hours for all cases (Table 5.2.4 and Figure 5.2.4). These results suggest that loading factor does not influence the file status much if the file has a long life time.

The distribution change of the number of CIs over the CI's was examined for the file whose CI capacity is 9. This experiment shows how the file structure changes over time for the given parameter values. At the file creation time, records are loaded with a loading factor 6; so, $X_6(0)$ is 8334 and $X_i(0)$ ($1 \le i \le 9$, $i \ne 6$) are zero. After the loading time, the number of CIs is spread over the other CI's as a result of records insertions and deletions and CI splits. Table 5.2.5 and Figure 5.2.5 show that, at time 90, $X_5(90)$ is even bigger than $X_6(90)$ and retains the bigger values over the entire time horizon. The reason for this is that $X_5(t)$ has an extra increasing source from the CI splits. As the time period becomes big enough, $X_i(t)$ ($1 \le i \le 4$) begin to grow significantly and, before time 400, $X_4(t)$ becomes even bigger than $X_8(t)$ and $X_9(t)$. The relative portion of each $X_i(t)$ is shown in Table 5.2.6 and Figure 5.2.6. Whereas weights of $X_i(t)$ ($5 \le i \le 9$) gradually decrease, those of $X_i(t)$ ($1 \le i \le 4$) increase over time. This situation indicates that

file utilization decreases with the increase in the number of CIs that contains a smaller number of records. The decline of file utilization is also supported in Figure 5.2.2. As illustrated earlier, utilization begins to drop after time 60 hours and becomes smaller than the initial utilization after time 300 hours. Both the increase in the number of total CIs and the decrease in utility become the evidence of file usage deterioration. The following chapters contain further discussion of file usage deterioration and its solutions.

Table 5.2.1 Growth of CIs for three different CI capacity sizes (CA9, CA15, CA 21)

Time	CA 9	CA 15	CA 21
0	8334	5000	3572
10	8336	5000	3572
20	8356	5002	3572
30	8414	5013	3575
40	8521	5044	3585
50	8677	5104	3610
60	8875	5195	3656
70	9106	5317	3726
80	9362	5464	3820
90	9633	5631	3934
100	9913	5812	4064
110	10200	6001	4206
120	10480	6193	4354
130	10760	6384	4505
140	11040	6573	4655
150	11310	6757	4802
160	11570	6935	4945
170	11840	7106	5081
180	12090	7271	5211
190	12340	7430	5335
200	12580	7582	5453
210	12820	7729	5564
220	13060	7871	5669
230	13290	8009	5770
240	13520	8142	5865
250	13750	8272	5957
260	13970	8400	6045
270	14190	8524	6129
280	14410	8646	6212
290	14630	8767	6292
300	14840	8886	6370
310	15060	9004	6448
320	15270	9120	6523
330	15480	9236	6598
340	15690	9351	6673
350	15900	9464	6746
360	16100	9578	6820
370	16310	9690	6893
380	16510	9802	6966
390	16710	9914	7038
400	16910	10020	7111
410	17110	10130	7183
420	17310	10240	7256
430	17510	10350	7328
440	17700	10460	7400
450	17900	10570	7472
460	18090	10680	7544
470	18290	10790	7616
480	18480	10890	7688
490	18670	11000	7760
500	18860	11100	7831

Figure 5.2.1 Growth of the CIs for three different CI capacity sizes

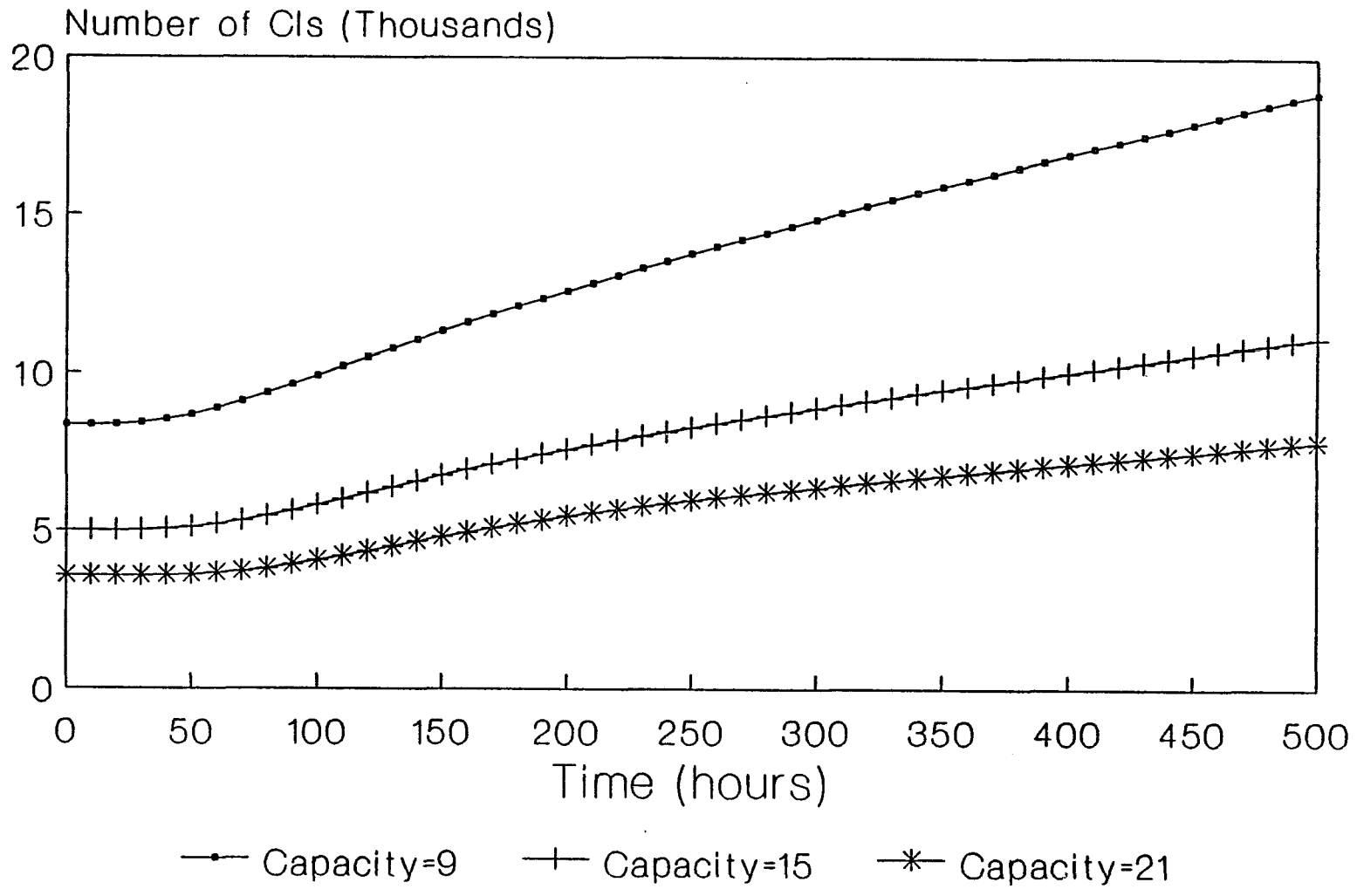


Table 5.2.2 Utility changes for three different CI capacity sizes (CA9, CA15, CA21)

Time	CA 9	CA 15	CA 21
0	0.6667	0.6667	0.6667
10	0.6864	0.6866	0.6865
20	0.7044	0.706	0.7061
30	0.7188	0.7239	0.7251
40	0.7287	0.7385	0.7423
50	0.734	0.7486	0.7561
60	0.7353	0.7537	0.7651
70	0.7338	0.7541	0.7687
80	0.7303	0.7507	0.7671
90	0.7257	0.7448	0.7615
100	0.7204	0.7373	0.753
110	0.7151	0.7291	0.743
120	0.7099	0.7209	0.7323
130	0.7051	0.7131	0.7218
140	0.7006	0.7059	0.7119
150	0.6965	0.6995	0.703
160	0.6928	0.6939	0.6951
170	0.6896	0.689	0.6883
180	0.6866	0.685	0.6826
190	0.684	0.6815	0.6779
200	0.6816	0.6786	0.6741
210	0.6794	0.6763	0.6711
220	0.6774	0.6743	0.6688
230	0.6755	0.6727	0.667
240	0.6738	0.6714	0.6658
250	0.6722	0.6703	0.6649
260	0.6707	0.6694	0.6644
270	0.6693	0.6686	0.6641
280	0.6679	0.6679	0.6641
290	0.6665	0.6673	0.6641
300	0.6652	0.6668	0.6643
310	0.664	0.6663	0.6646
320	0.6628	0.6657	0.6648
330	0.6615	0.6652	0.6651
340	0.6603	0.6647	0.6654
350	0.6591	0.6642	0.6656
360	0.658	0.6637	0.6658
370	0.6568	0.6631	0.6659
380	0.6556	0.6626	0.666
390	0.6545	0.662	0.666
400	0.6534	0.6614	0.666
410	0.6522	0.6608	0.6659
420	0.6511	0.6601	0.6658
430	0.65	0.6595	0.6656
440	0.6489	0.6588	0.6653
450	0.6478	0.6581	0.665
460	0.6467	0.6574	0.6647
470	0.6456	0.6567	0.6643
480	0.6445	0.656	0.6639
490	0.6434	0.6553	0.6634
500	0.6423	0.6545	0.6629

Figure 5.2.2 Utility changes for three different CI capacity sizes

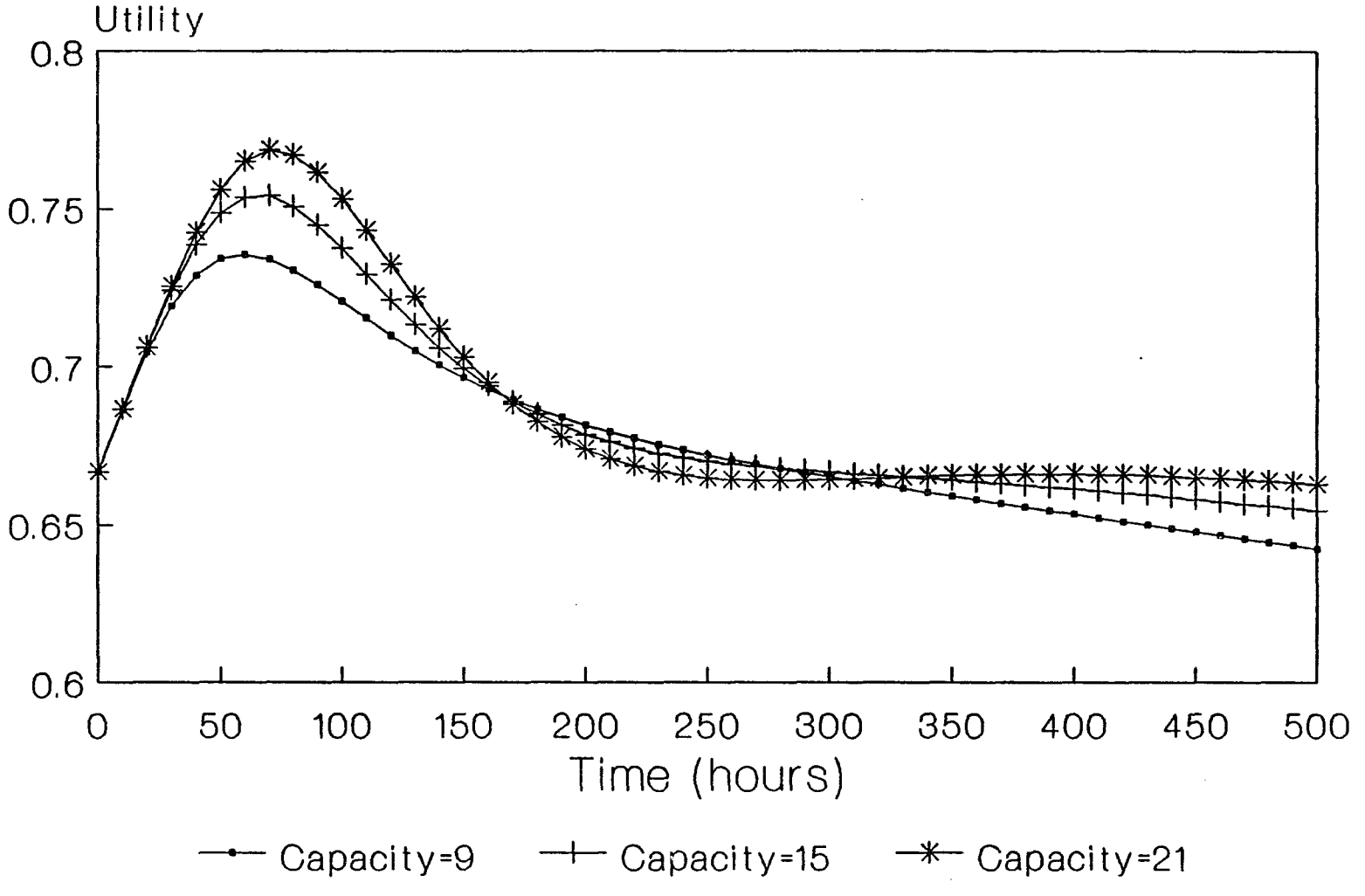


Table 5.2.3 Effect of loading factors on the total number of CIs in a file

Time	L=9	L=10	L=11	L=12	L=13	L=14
0	5556	5000	4546	4167	3847	3572
10	5556	5000	4547	4175	3912	3943
20	5556	5002	4560	4244	4165	4547
30	5558	5013	4609	4403	4532	5088
40	5566	5044	4708	4638	4927	5515
50	5584	5104	4857	4917	5301	5843
60	5618	5195	5046	5212	5633	6092
70	5671	5317	5262	5502	5919	6285
80	5745	5464	5492	5777	6162	6438
90	5838	5631	5726	6031	6370	6563
100	5950	5812	5957	6261	6547	6670
110	6077	6001	6180	6489	6702	6766
120	6216	6193	6393	6658	6838	6855
130	6365	6384	6593	6829	6962	6942
140	6520	6573	6781	6985	7076	7030
150	6679	6757	6957	7128	7185	7119
160	6839	6935	7122	7263	7290	7212
170	7000	7106	7278	7390	7394	7309
180	7160	7271	7426	7511	7497	7410
190	7318	7430	7566	7629	7601	7515
200	7473	7582	7700	7744	7706	7624
210	7625	7729	7830	7857	7813	7736
220	7774	7871	7956	7969	7921	7851
230	7919	8009	8078	8081	8032	7969
240	8061	8142	8199	8192	8144	8088
250	8200	8272	8317	8304	8258	8209
260	8336	8400	8434	8417	8373	8332
270	8469	8524	8550	8529	8489	8454
280	8599	8646	8665	8643	8606	8578
290	8727	8767	8779	8756	8724	8701
300	8852	8886	8894	8870	8842	8824
310	8976	9004	9008	8985	8960	8947
320	9098	9120	9121	9100	9079	9070
330	9218	9236	9234	9214	9197	9191
340	9337	9351	9347	9329	9315	9312
350	9454	9464	9460	9444	9433	9432
360	9570	9578	9573	9558	9550	9551
370	9685	9690	9685	9673	9667	9670
380	9800	9802	9797	9786	9783	9787
390	9913	9914	9909	9900	9898	9903
400	10020	10020	10020	10010	10010	10020
410	10140	10130	10130	10130	10130	10130
420	10250	10240	10240	10240	10240	10250
430	10360	10350	10350	10350	10350	10360
440	10470	10460	10460	10460	10460	10470
450	10570	10570	10570	10570	10570	10580
460	10680	10680	10680	10680	10680	10690
470	10790	10790	10790	10790	10790	10800
480	10900	10890	10890	10900	10900	10910
490	11000	11000	11000	11000	11000	11020
500	11110	11100	11110	11110	11110	11120

Figure 5.2.3 Effect of loading factors on the total number of CIs in a file

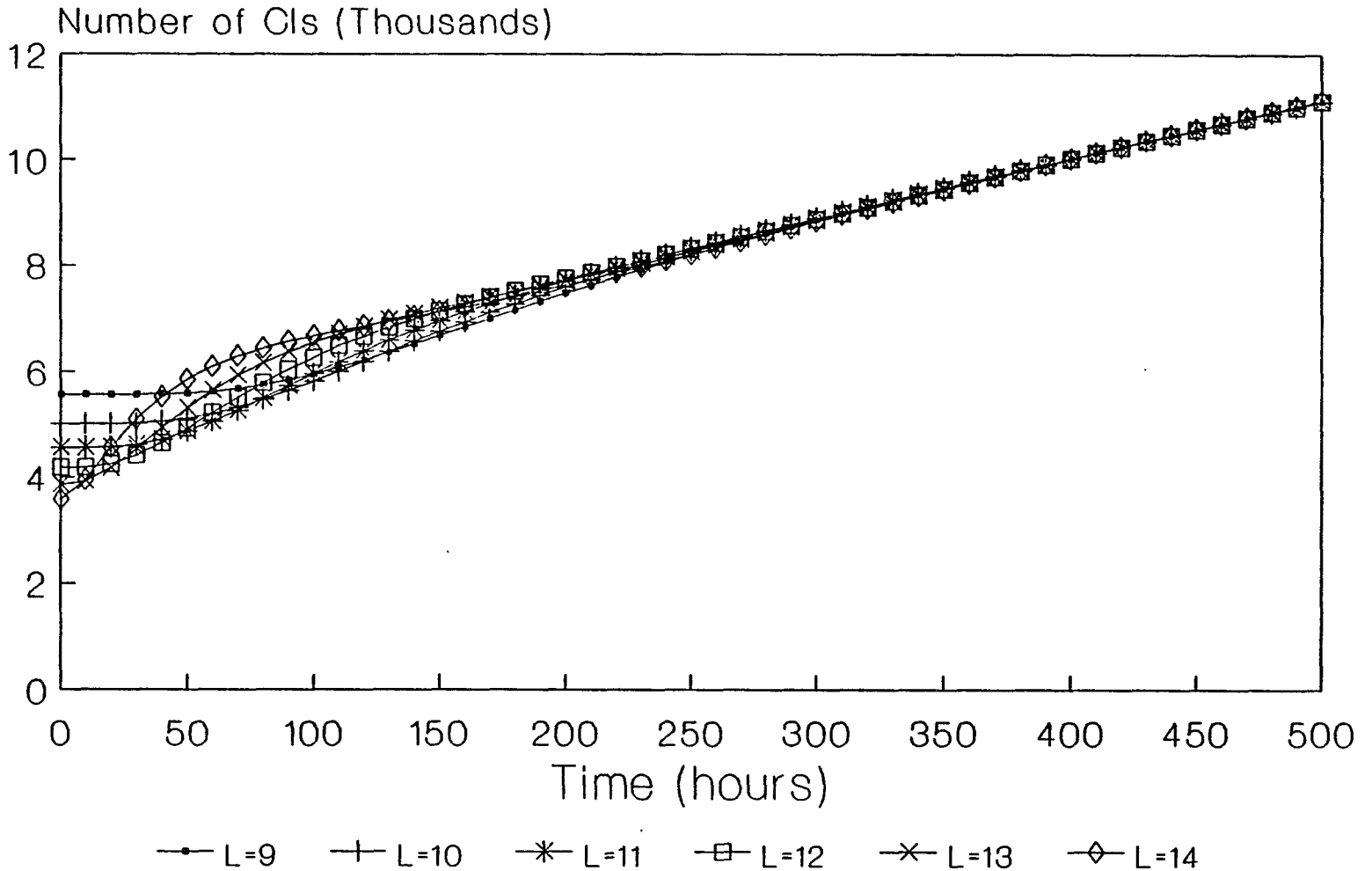


Table 5.2.4 Effect of loading factors on the utility of a file

Time	L=9	L=10	L=11	L=12	L=13	L=14
0	0.6	0.6667	0.7332	0.7999	0.8665	0.9332
10	0.6179	0.6866	0.755	0.8222	0.8775	0.8706
20	0.6356	0.706	0.7744	0.8322	0.8479	0.7767
30	0.6529	0.7239	0.7873	0.8242	0.8006	0.7133
40	0.6694	0.7385	0.7913	0.8032	0.7561	0.6755
50	0.6843	0.7486	0.7867	0.7771	0.7208	0.654
60	0.697	0.7537	0.776	0.7513	0.6952	0.6427
70	0.707	0.7541	0.7619	0.7287	0.6774	0.6379
80	0.7141	0.7507	0.7469	0.7101	0.6657	0.6371
90	0.7184	0.7448	0.7324	0.6954	0.6584	0.639
100	0.7202	0.7373	0.7193	0.6844	0.6545	0.6424
110	0.72	0.7291	0.7079	0.6762	0.6528	0.6466
120	0.7182	0.7209	0.6983	0.6705	0.6528	0.6512
130	0.7153	0.7131	0.6905	0.6667	0.6539	0.6557
140	0.7116	0.7059	0.6843	0.6643	0.6557	0.66
150	0.7077	0.6995	0.6794	0.663	0.6578	0.6638
160	0.7035	0.6939	0.6756	0.6625	0.66	0.6672
170	0.6995	0.689	0.6728	0.6626	0.6623	0.6699
180	0.6956	0.685	0.6707	0.6631	0.6643	0.6721
190	0.6919	0.6815	0.6693	0.6637	0.6662	0.6738
200	0.6886	0.6786	0.6682	0.6645	0.6678	0.6749
210	0.6855	0.6763	0.6676	0.6653	0.6689	0.6757
220	0.6828	0.6743	0.6672	0.6661	0.6701	0.6761
230	0.6804	0.6727	0.6669	0.6668	0.6708	0.6761
240	0.6782	0.6714	0.6668	0.6673	0.6713	0.6759
250	0.6762	0.6703	0.6667	0.6677	0.6715	0.6754
260	0.6745	0.6694	0.6667	0.668	0.6715	0.6748
270	0.673	0.6686	0.6666	0.6682	0.6714	0.6741
280	0.6716	0.6679	0.6665	0.6682	0.6711	0.6733
290	0.6704	0.6673	0.6664	0.6681	0.6706	0.6724
300	0.6693	0.6668	0.6662	0.6679	0.6701	0.6714
310	0.6683	0.6663	0.666	0.6676	0.6695	0.6705
320	0.6674	0.6657	0.6657	0.6673	0.6688	0.6695
330	0.6665	0.6652	0.6653	0.6668	0.668	0.6685
340	0.6657	0.6647	0.6649	0.6662	0.6673	0.6675
350	0.6649	0.6642	0.6645	0.6657	0.6664	0.6665
360	0.6642	0.6637	0.664	0.665	0.6656	0.6655
370	0.6635	0.6631	0.6635	0.6643	0.6647	0.6646
380	0.6628	0.6626	0.6629	0.6636	0.6639	0.6636
390	0.6621	0.662	0.6623	0.6629	0.663	0.6627
400	0.6614	0.6614	0.6617	0.6622	0.6622	0.6618
410	0.6607	0.6608	0.661	0.6614	0.6613	0.6609
420	0.66	0.6601	0.6603	0.6606	0.6604	0.66
430	0.6593	0.6595	0.6596	0.6598	0.6596	0.6592
440	0.6586	0.6588	0.6589	0.659	0.6587	0.6583
450	0.6579	0.6581	0.6582	0.6582	0.6579	0.6575
460	0.6572	0.6574	0.6574	0.6574	0.6571	0.6567
470	0.6565	0.6567	0.6567	0.6566	0.6563	0.6559
480	0.6558	0.656	0.6559	0.6558	0.6554	0.6551
490	0.6551	0.6553	0.6551	0.655	0.6546	0.6543
500	0.6544	0.6545	0.6544	0.6542	0.6538	0.6535

Figure 5.2.4 Effect of loading factors on the utility of a file

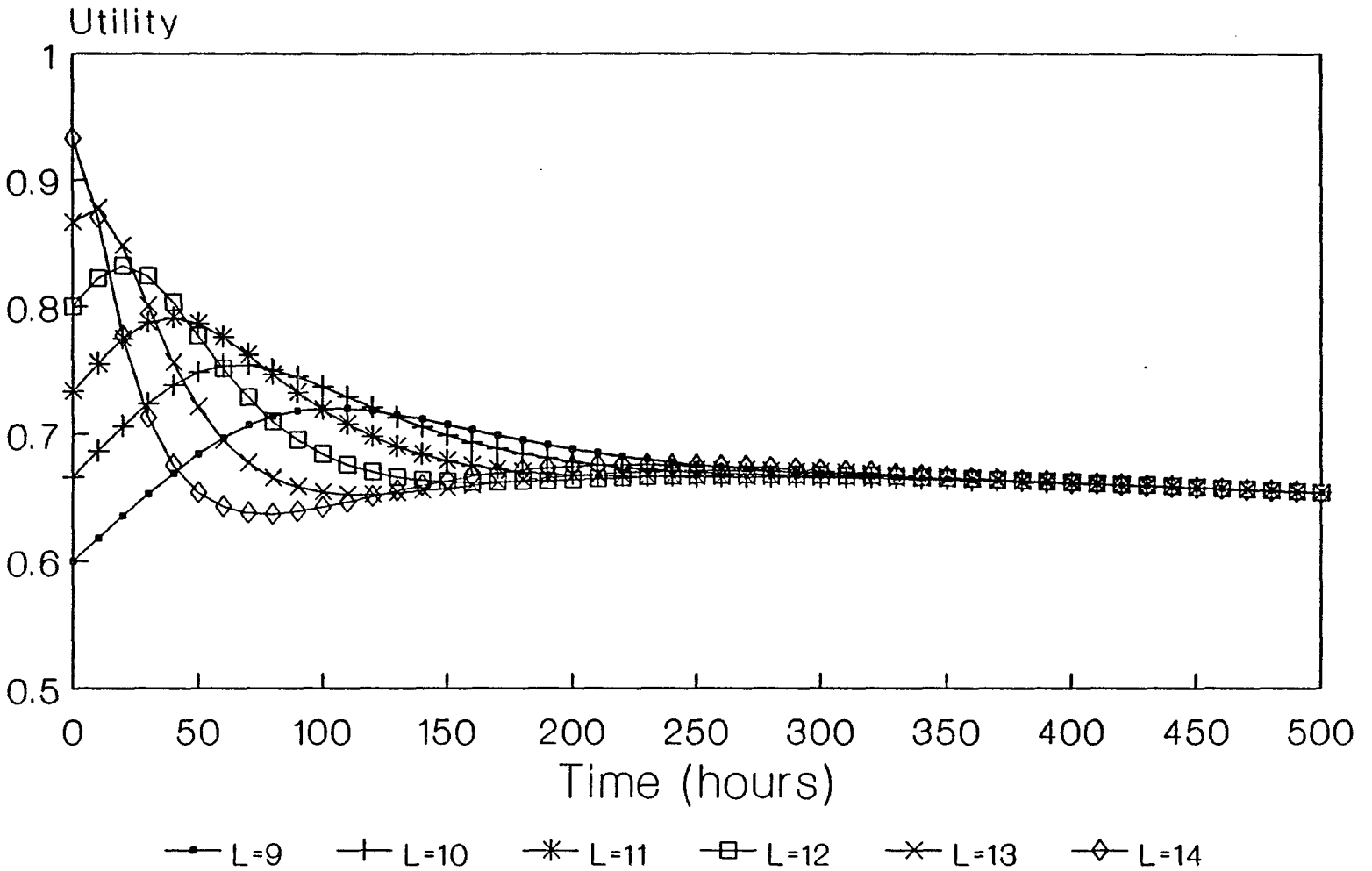


Table 5.2.5 CI spreading pattern over the entire CI's after initial loading

Time	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	8334	0	0	0
10	0	0	0	10	387	6284	1442	194	20
20	0	0	1	32	638	4903	2128	547	107
30	0	0	2	61	849	3954	2411	888	248
40	0	0	5	95	1070	3299	2482	1161	410
50	0	0	9	134	1311	2849	2448	1357	568
60	0	1	14	180	1570	2548	2368	1489	706
70	0	1	20	231	1836	2357	2275	1569	818
80	0	2	27	287	2097	2246	2186	1613	904
90	0	2	36	348	2344	2193	2110	1633	967
100	0	3	46	412	2573	2182	2050	1637	1009
110	0	5	58	478	2780	2200	2006	1633	1036
120	0	6	71	545	2965	2239	1978	1624	1051
130	0	8	85	613	3128	2290	1964	1614	1059
140	1	10	101	680	3271	2349	1961	1605	1060
150	1	12	117	746	3397	2412	1967	1599	1058
160	1	15	135	811	3507	2476	1980	1595	1054
170	1	18	153	875	3604	2540	1999	1594	1050
180	2	22	173	936	3690	2602	2023	1597	1046
190	2	26	193	996	3768	2662	2049	1601	1042
200	3	30	213	1054	3839	2720	2078	1608	1039
210	3	34	234	1110	3904	2776	2108	1618	1037
220	4	39	255	1165	3965	2829	2138	1629	1036
230	4	44	277	1218	4023	2879	2169	1641	1037
240	5	50	299	1270	4078	2927	2200	1654	1038
250	6	56	322	1320	4131	2974	2231	1669	1040
260	7	62	345	1370	4128	3018	2262	1684	1042
270	8	68	368	1418	4233	3061	2292	1699	1046
280	9	75	391	1466	4282	3103	2321	1715	1050
290	10	82	414	1513	4331	3143	2350	1731	1054
300	12	90	438	1559	4380	3182	2379	1746	1059
310	13	97	462	1604	4428	3220	2407	1762	1063
320	15	105	486	1650	4476	3258	2434	1778	1068
330	17	114	510	1694	4523	3295	2461	1794	1073
340	18	122	534	1739	4570	3331	2487	1809	1079
350	20	131	559	1783	4617	3366	2512	1824	1084
360	22	140	583	1326	4663	3401	2538	1839	1089
370	24	150	608	1870	4709	3435	2562	1854	1094
380	27	159	633	1913	4755	3469	2587	1868	1099
390	29	169	658	1956	4800	3503	2610	1882	1104
400	32	180	684	1999	4845	3536	2634	1896	1108
410	34	190	709	2041	4890	3568	2657	1910	1113
420	37	201	735	2084	4934	3600	2679	1923	1118
430	40	212	761	2126	4977	3632	2702	1936	1122
440	43	223	787	2169	5021	3664	2723	1949	1126
450	47	234	813	2211	5064	3695	2745	1961	1130
460	50	246	839	2253	5106	3725	2766	1974	1134
470	54	258	866	2294	5148	3756	2787	1986	1138
480	57	270	892	2336	5190	3786	2808	1997	1142
490	61	283	919	2378	5231	3815	2828	2009	1145
500	65	296	646	2419	5271	3844	2848	2020	1149

Figure 5.2.5 CI spreading pattern over the entire CI's after initial loading

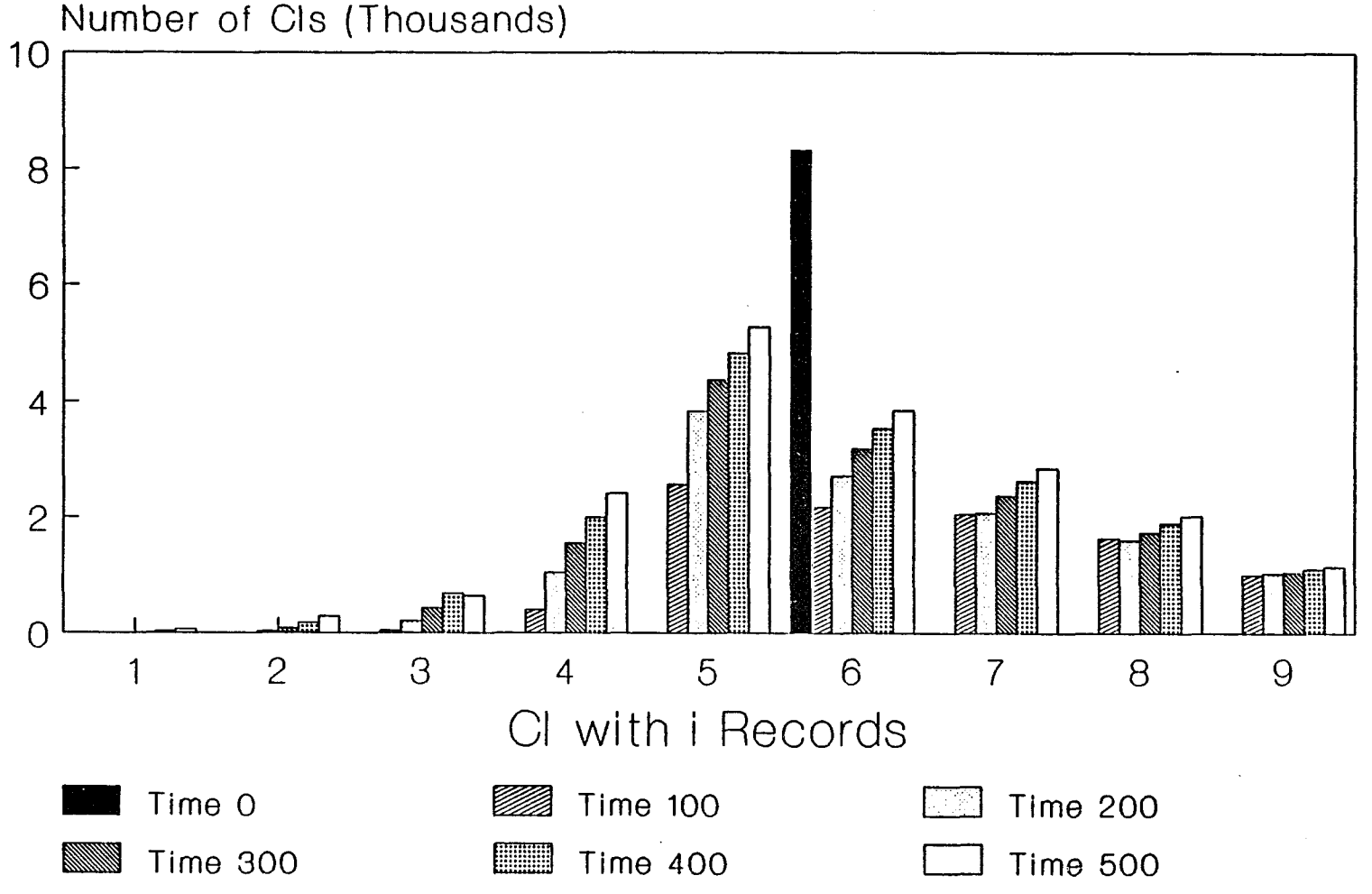
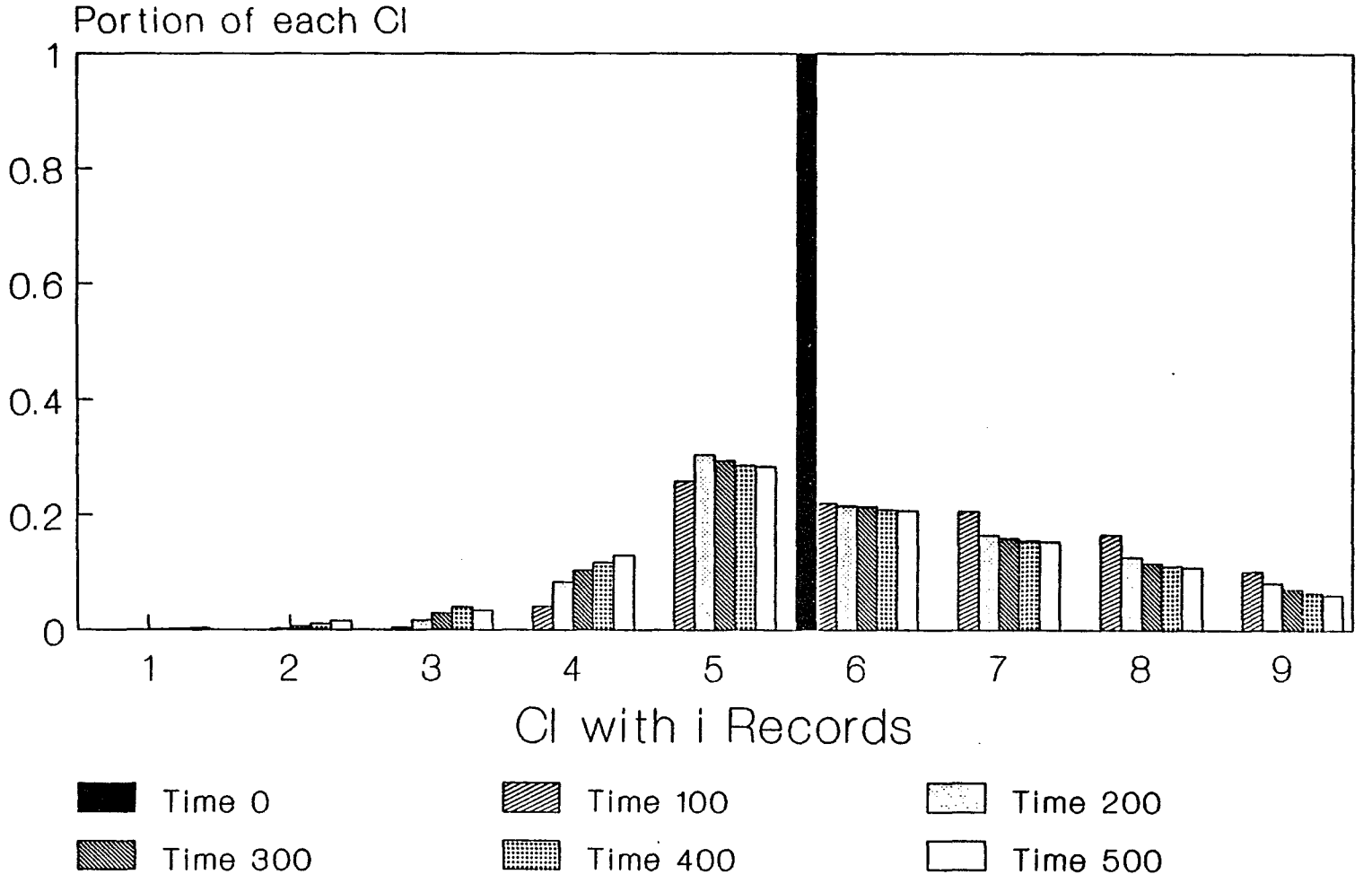


Table 5.2.6 The relative portion of each C_i^i

Time	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0
10	0	0	0	0.0011	0.0464	0.7537	0.1729	0.0232	0.0023
20	0	0	0.0001	0.0038	0.0763	0.5867	0.2546	0.0654	0.0128
30	0	0	0.0002	0.0072	0.1009	0.4699	0.2865	0.1055	0.0294
40	0	0	0.0005	0.0111	0.1255	0.3871	0.2912	0.1362	0.0481
50	0	0	0.0010	0.0154	0.1511	0.3283	0.2821	0.1564	0.0654
60	0	0.0001	0.0015	0.0202	0.1768	0.2870	0.2667	0.1677	0.0795
70	0	0.0001	0.0021	0.0253	0.2016	0.2588	0.2498	0.1722	0.0898
80	0	0.0002	0.0028	0.0306	0.2239	0.2399	0.2334	0.1722	0.0965
90	0	0.0002	0.0037	0.0361	0.2433	0.2276	0.2190	0.1695	0.1003
100	0	0.0003	0.0046	0.0415	0.2595	0.2201	0.2068	0.1651	0.1017
110	0	0.0004	0.0056	0.0468	0.2726	0.2157	0.1967	0.1601	0.1016
120	0	0.0005	0.0067	0.0520	0.2829	0.2136	0.1887	0.1549	0.1002
130	0	0.0007	0.0078	0.0569	0.2906	0.2128	0.1825	0.1499	0.0984
140	0.0000	0.0009	0.0091	0.0616	0.2963	0.2128	0.1776	0.1454	0.0960
150	0.0000	0.0010	0.0103	0.0659	0.3003	0.2132	0.1739	0.1413	0.0935
160	0.0000	0.0012	0.0116	0.0700	0.3030	0.2139	0.1710	0.1378	0.0910
170	0.0000	0.0015	0.0129	0.0739	0.3045	0.2146	0.1689	0.1346	0.0887
180	0.0001	0.0018	0.0143	0.0774	0.3051	0.2152	0.1673	0.1320	0.0865
190	0.0001	0.0021	0.0156	0.0807	0.3053	0.2157	0.1660	0.1297	0.0844
200	0.0002	0.0023	0.0169	0.0837	0.3050	0.2161	0.1651	0.1277	0.0825
210	0.0002	0.0026	0.0182	0.0865	0.3044	0.2164	0.1643	0.1261	0.0808
220	0.0003	0.0029	0.0195	0.0892	0.3035	0.2166	0.1637	0.1247	0.0793
230	0.0003	0.0033	0.0208	0.0916	0.3026	0.2165	0.1631	0.1234	0.0780
240	0.0003	0.0036	0.0221	0.0939	0.3016	0.2164	0.1627	0.1223	0.0767
250	0.0004	0.0040	0.0234	0.0960	0.3004	0.2163	0.1622	0.1213	0.0756
260	0.0005	0.0044	0.0247	0.0984	0.2965	0.2168	0.1625	0.1209	0.0748
270	0.0005	0.0047	0.0259	0.0999	0.2982	0.2156	0.1614	0.1197	0.0736
280	0.0006	0.0052	0.0271	0.1017	0.2971	0.2153	0.1610	0.1189	0.0728
290	0.0006	0.0056	0.0283	0.1034	0.2960	0.2148	0.1606	0.1183	0.0720
300	0.0008	0.0060	0.0295	0.1050	0.2950	0.2143	0.1602	0.1176	0.0713
310	0.0008	0.0064	0.0306	0.1065	0.2941	0.2138	0.1598	0.1170	0.0706
320	0.0009	0.0068	0.0318	0.1080	0.2931	0.2133	0.1593	0.1164	0.0699
330	0.0010	0.0073	0.0329	0.1094	0.2921	0.2128	0.1589	0.1158	0.0693
340	0.0011	0.0077	0.0340	0.1108	0.2912	0.2123	0.1585	0.1153	0.0687
350	0.0012	0.0082	0.0351	0.1121	0.2904	0.2117	0.1580	0.1147	0.0681
360	0.0013	0.0086	0.0362	0.1134	0.2896	0.2112	0.1576	0.1142	0.0676
370	0.0014	0.0091	0.0372	0.1146	0.2887	0.2106	0.1571	0.1137	0.0670
380	0.0016	0.0096	0.0383	0.1158	0.2880	0.2101	0.1566	0.1131	0.0665
390	0.0017	0.0101	0.0393	0.1170	0.2872	0.2096	0.1561	0.1126	0.0660
400	0.0018	0.0106	0.0404	0.1181	0.2864	0.2090	0.1557	0.1120	0.0655
410	0.0019	0.0111	0.0414	0.1192	0.2857	0.2085	0.1552	0.1116	0.0650
420	0.0021	0.0116	0.0424	0.1203	0.2850	0.2079	0.1547	0.1110	0.0645
430	0.0022	0.0121	0.0434	0.1214	0.2842	0.2074	0.1543	0.1105	0.0640
440	0.0024	0.0125	0.0444	0.1225	0.2835	0.2069	0.1537	0.1100	0.0635
450	0.0026	0.0130	0.0454	0.1235	0.2829	0.2064	0.1533	0.1095	0.0631
460	0.0027	0.0135	0.0463	0.1245	0.2822	0.2058	0.1528	0.1091	0.0626
470	0.0029	0.0141	0.0473	0.1254	0.2815	0.2053	0.1524	0.1086	0.0622
480	0.0030	0.0146	0.0482	0.1264	0.2808	0.2048	0.1519	0.1080	0.0618
490	0.0032	0.0151	0.0492	0.1273	0.2801	0.2043	0.1514	0.1076	0.0613
500	0.0035	0.0159	0.0348	0.1303	0.2840	0.2071	0.1534	0.1088	0.0619

Figure 5.2.6 The relative portion of each CI^i



Chapter 6

PHYSICAL DATABASE REORGANIZATION

The discussion in Section 4.1 focused on the file behavior under the assumption that a file is built only through continuous records insertions. For a large number of records, the probability of an insertion's occurring in each CI was obtained. In Section 4.2, a formulation of the expected number of CIs was obtained by considering the insertions and the deletions of records together. The birth and death process was used to describe the transitional behavior of fringes. In this chapter, we consider only the model developed in Section 4.2 for further discussion. The main purposes of this chapter are to describe the file deterioration through the analysis of file operation and to set up the optimal reorganization policy.

6.1. Analysis of File Operation Cost on Behalf of CA Splits

In a file such as a VSAM file, that permits distributed free space, two records access methods are considered. One is a direct access, and the other is a sequential access. The difference between the two is well stated in Maruyama and Smith (1976). Direct accessing of records by their keys depends on the index structure of a file. VSAM is an indexed-sequential file organization. It permits multiple

secondary indexes. The indexes have a B^+-tree structure (Comer, 1979). The sequential access of records depends on the location of the records. If a record currently accessed and a record that is to be accessed next are in the same CI, then the next record can be accessed directly. If it is, however, in the next CI within the same CA as the current CI, then it is retrieved sequentially, using the entries in the sequence set index since the entries are in sequence. If the next CI is in a different CA, then the horizontal pointer in the CA set must be traversed to get the sequence set index for that CA, and then the next CI can be accessed using the vertical pointer in the sequence set entry. Sequential accesses over CA boundaries are significantly more costly than those that are not, since they require mechanical movement of Read/Write heads. The cost of sequential accesses is a function of the distance that the Read/Write heads must move to get to the next CA.
In other words, the size of a file is denoted by a set of CAs, and the cost of accessing a file depends on the number of CAs. The file growing pattern through the CI and CA splits is described in the following way. Until the file growth is insufficient to fill the free CIs within a CA after the file is loaded or reorganized, no CA split occurs and the cost of accessing the next logical CA in key sequence remains constant. As soon as the number of CI splits exceeds free space in a CA, the CA split begins.

When a CA split occurs, a new CA obtains free space at the end of the file, making logically close CA to be physically far apart. This causes the additional increase in the cost of accessing the next logical CA. This cost could be reduced by allocating the optimal size of free spaces in the file. Free spaces reduce the probability of CI and CA splits and, as a result, improve file performance. This, in turn, reduces the likelihood of moving a set of records to a different CA away from the other records that have been in the key sequence and reduces the sequential access cost. However, too much or too little free space is not desirable. Too much free space requires more I/O operations to do sequential processing of the same number of records. Too little space causes the frequent splits of CI and CA. It causes an excessive number of CIs and CAs and results in an excessive time requirement for sequential processing because of the disorder of the physical sequence of the data (IBM Corp., 1985). The size of free space is decided by a loading factor. The loading factor should be decided in the way of maximizing file performance or minimizing file execution cost by preventing wasted space as well as too frequent CA splits. A CA split requires 248 I/O's, for example, on an IBM 3380, if a CI capacity size is 4K (Ranade, 1987). At the rate of 12 I/O's per second, it would take about 20.6 seconds to execute 248 I/O's. This is a considerable time to wait in an on-line environment.

In Section 4.2, we calculated the total expected number of CIs at time t, using the simulation method. However, we did not consider the relationship between CIs and CAs in that section. The relationship between them and its effect on file operation costs can be described in the following way.

Let m be the maximum number of CIs in a CA and f be the number of free CIs allocated in a CA. The first CA split would not occur at least until the growth of the file causes the f+1 times of CI splits. If more than f times of CI splits occur (i.e., if F(t) - F(0) exceeds f in a CA), the CA split can be considered. The initial number of CAs, Z(0), is defined as Z(0) = F(0)/(m-f), and the maximum number of CA splits is calculated by (F(t) - F(0))/(f + 1). Therefore, the maximum number of CAs at time t, Z(t), is given by

$$Z(t) = Z(0) + (F(t) - F(0)) / (f + 1). \qquad (6.1.1)$$

As explained earlier in this section, the cost of accessing a file depends on Z(t). Let the cost of accessing a CA before a split be h(0) and the cost at time t be h(t). The cost h(t) will increase as more of the costly splits occur, until it reaches a maximum value h_{max}. Because of the capacity limit of a file, the number of CAs in a file is limited to a finite number. Let define Z_m be the maximum number of CAs in a file. Then, the maximum accessing cost

for this finite number of CAs will be h_{max}. If the file operation requires more than Z_m CAs as a result of CA splits, then the system is locked. If the system is locked, we cannot do any operations on that file. In this case, since we can no longer use the present file, we need to retrieve a backup file. To do this, we must execute the proper file operation again. In this undesirable situation, the accessing cost could be considered infinite. Since the cost of accessing the next logical CA is bounded by the cost of accessing the first CA and the last CA of a file, this cost can be represented in the following way:

$$h(t) = h(0), \qquad \text{for } INT(Z(t)) = Z(0)$$

$$h(t) = h(0) + \epsilon\,(Z(t) - Z(0)) \,/\, (Z_m + 1 - Z(t)), \qquad \text{for } INT(Z(t)) > Z(0)$$

$$h(t) = \infty, \qquad \text{for } INT(Z(t)) > Z_m, \qquad (6.1.2)$$

where ϵ is the deterioration rate.

As data are inserted, deleted, and updated, the structure of the physical database or file is deteriorated, and this deterioration causes inefficient data or records retrievals. This, in turn, results in the excess costs of file operation. The operation cost of a file is based only on the retrieval and update queries for the database that cause sequential CAs accesses. The addition and deletion costs are negligible. Because these operations are direct, the performance of these operations is not affected by the physical disarrangement of the file. Therefore, only

retrievals and updates are considered in this study, and they are treated in the same way. The total excess costs caused by the access of CAs during a time interval are given by

$$TAC = \int_0^t \gamma\, g(s)\ (h(t') - h(0))\ dt', \qquad (6.1.3)$$

where $\gamma g(s)$ is the sequential CA access rate due to retrievals, γ is the query retrieval and update rate at a given time period and assumed to be constant during that period, and $g(s)$ is the expected number of sequential CA accesses for a query that requests s records accesses in sequence.

When the pointer chains become long as a result of splits, retrieval and update operations in a sequential processing involve excessive time cost. Too much excessive time cost is not desirable for the sake of smooth and efficient operations. In such a case, a reorganization is requested to regulate this excessive cost.

6.2. Physical Database Reorganization Policy

Reorganization of a database is usually managed by the database administrator (DBA) (Teorey and Fry, 1982). The DBA determines that reorganization should be performed, decides when it is best done, and carries out the

implementation with DBMS. The issues considered by the DBA in carrying out a reorganization are well described in Sockut and Goldberg (1979). These issues include:

(1) Deciding what new structures are to be the final result of reorganization.

(2) Deciding when to perform reorganization.

(3) Knowing how to execute the reorganization.

(4) Accessing how much reorganization will cost.

Reorganization of the database could be performed at a fixed time, at the end of the n^{th} time period, or at the time when the database is deteriorated to some point after the most recent reorganization. In general, however, the reorganization point may be determined by comparing total excess cost and reorganization cost. The optimal reorganization policy minimizes the total operating costs by specifying when to reorganize a file. The policy balances tradeoff between the extra cost of reorganization and its benefit in restoring file performance efficiency and reducing the excess cost of ensuing transactions.

Reorganization is performed by unloading (reading) the file sequentially and reloading (writing) the records sequentially into the newly restructured file. During this process, new indexes are created based on the changed distribution of key values. The basic assumption for the estimate of the reorganization cost is that this cost is a function of the size of a file. Under this assumption, the

cost of j^{th} reorganization is given by the following equation (Maruyama and Smith (1976)):

$$R_j(t) = R_0 + \frac{\lambda\ r\ Z_j(t)}{\xi(m-f)\ \gamma_j\ (Z_j(t)-Z_j(0))}\ , \qquad (6.2.1)$$

where R_0 is constant and represents the overhead cost associated with reorganizing the file,

$\xi(m-f)$ is the space utilization within a CA, and

r is the sum of the time required to read a full CA, plus the time to write a full CA.

The total operating cost at the j^{th} reorganization period is given by $TC_j(t) = TAC_j(t) + R_j(t)$. Since the optimal reorganization policy minimizes the total cost, we differentiate $TC_j(t)$ for $Z_j(t)$ and find the point that makes this differentiation zero. This scheme gives the optimal reorganization point in terms of $Z_j(t)$.

$$\frac{dTC_j(t)}{dZ_j(t)} = \frac{dTAC_j(t)}{dZ_j(t)} + \frac{dR_j(t)}{dZ_j(t)} \qquad (6.2.2)$$

Considering the first term on the right-hand side first,

$$\frac{dTAC_j(t)}{dZ_j(t)} = \left(\frac{dTAC_j(t)}{dt}\right)\left(\frac{dt}{dZ_j(t)}\right)$$

$$= (\gamma_j g(s)\ (h_j(t) - h_j(0)))\ (dt/dZ_j(t)), \qquad (6.2.3)$$

we can get $dt/dZ_j(t)$ as an inverse of $dZ_j(t)/dt$. From equation (6.1.1), we can get

$$Z_j(t) = Z_j(0) + (F_j(t) - F_j(0)) / (f + 1). \qquad (6.2.4)$$

From equation (6.2.4), we have

$$\frac{dZ_j(t)}{dt} = \frac{1}{f+1}\frac{dF_j(t)}{dt}$$

$$= \frac{1}{f+1}\frac{d}{dt}\left(\sum_{i=1}^{b} Y_{i,j}(t)\right) = \frac{1}{f+1}\sum_{i=1}^{b}\frac{d}{dt} Y_{i,j}(t).$$

Using equations (4.2.11) through (4.2.14), we can derive following relation:

$$\sum_{i=1}^{b}\frac{d}{dt} Y_i(t) = \alpha \sum_{i=2}^{b} (i-1)\ Y_{i-1}(t) - (\alpha+\mu) \sum_{i=1}^{b} i\ Y_i(t)$$

$$+ \mu \sum_{i=1}^{b-1} (i+1)\ Y_{i+1}(t) + 2\alpha b\ Y_b(t)$$

$$= -\mu\ Y_1(t) + \alpha b\ Y_b(t).$$

Therefore, we have

$$\frac{dZ_j(t)}{dt} = \frac{-\mu\ Y_{1,j}(t) + \alpha b\ Y_{b,j}(t)}{f+1}. \qquad (6.2.5)$$

Let us put $\Gamma_j(t) \equiv dZ_j(t)/dt$. Then, from equations (6.2.3) and (6.2.5), we have

$$dTAC_j(t)/dZ_j(t) = \gamma_j g(s)\ \Gamma_j(t)^{-1}\ (h_j(t) - h_j(0)). \qquad (6.2.6)$$

Differentiating equation (6.2.1) for $Z_j(t)$ gives

$$\frac{dR_j(t)}{dZ_j(t)} = \frac{-\lambda\, r\, Z_j(0)}{b(m-f)\ \gamma_j\ (Z_j(t)-Z_j(0))^2}\ . \qquad (6.2.7)$$

Therefore, the condition that the differentiation of total cost for $Z_j(t)$ equals zero, $dTC_j(t)/dZ_j(t) = 0$, gives the following relation:

$$\gamma_j g(s)\ \Gamma_j(t)^{-1}\ \epsilon\ (Z_j(t) - Z_j(0))\ /\ (Z_m + 1 - Z_j(t))$$
$$-\ \lambda\ r\ Z_j(0)\ /\ [\xi(m-f)\ \gamma_j\ (Z_j(t) - Z_j(0))^2] = 0. \qquad (6.2.8)$$

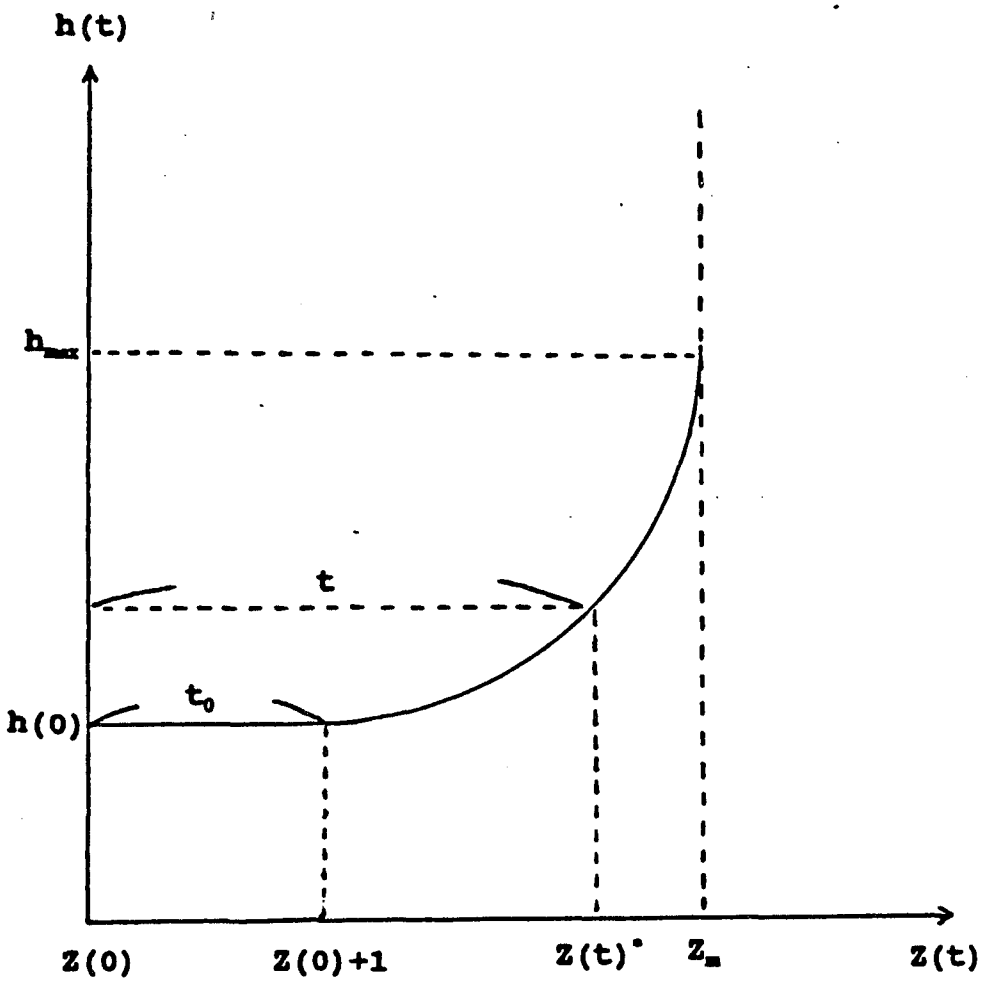
If we rearrange this equation, we have

$$D_j(t)\ (Z_j(t) - Z_j(0))^3 + Z_j(t) - Z_m - 1 = 0, \qquad (6.2.9)$$

where $D_j(t) = \gamma_j^2 g(s)\ \Gamma_j(t)^{-1}\ \epsilon\ \xi(m-f)\ /\ (\lambda\ r\ Z_j(0))$.

The solution of the equation (6.2.9), $Z_j(t)^*$, gives the optimal reorganization point. This means that when the number of CAs is $Z_j(t)^*$, reorganization is called. Figure 6.1.1 shows this situation graphically together with the cost function in equation (6.1.2). Since the equation (6.2.9) cannot be solved analytically, the simulation package SLAM II is used to obtain the solution numerically. A detailed analysis of the optimal reorganization point is discussed in the next chapter, using various parameter values adopted in IBM 3380.

Figure 6.1.1 Cost of accessing next logical CA sequentially



Chapter 7

NUMERICAL EXPERIMENTS FOR THE OPTIMAL REORGANIZATION POINT

The previous chapter described how the CAs are increased, based on the model discussed in Section 4.2. When the number of CAs increased as a result of CA splits, the database accessing cost per query also increased because CA splits disperse logically adjacent data to physically far apart locations. This cost function is given in equation (6.1.2). Based on this cost function and the reorganization cost given in equation (6.2.1), optimal policy was set up for reorganization of the database for each reorganization period.

In this chapter, we discuss the computational results of this optimal database reorganization policy. This policy is determined from equation (6.2.9) and depends on the various parameter values. For the analytical purpose, we consider IBM 3380, which is one of the IBM direct access storage devices (DASD). IBM 3380 gives the following characteristics (Ranade, 1987; IBM Corp., 1985).

Cylinders per diskpack	885
Tracks per cylinder	15
Bytes per track	47,968

Seek time delay (ms)	
minimum	3
average	16
maximum	30
Rotational delay (ms)	
average	8.4
maximum	16.8
Data transfer rate (MBS)	3
Data transfer delay for 4K CI (ms)	1.3
Number of I/O's on a CA split	248

For IBM 3380, if we assume that the size of a CA is one cylinder, and the size of CI is 4K, the number of data CIs per CA is given by 150; i.e., m = 150. Therefore, with this m value, if we choose about 10% free space in a CA, f will be 15. Also, if we assume about 80% loading of records in a CI, loading factor ξ will be 12 for given b=15.

The cost per cylinder to reorganize the file, r, is taken to be the sum of three terms: seek time delay, rotational delay, and data transfer delay. The following three steps are required to perform a successful I/O. First, the read-write arms search the desired cylinder. Second, the disk rotates until the read-write head is located on the desired track. Third, data is transferred from or to DASD. The values used in this analysis are selected in the following way. The maximum seek time, on

the average, is 30 ms. However, from the fact that a seek does not traverse more than one-third of the cylinders, we use an average seek time of 16 ms, instead of 30 ms. For the IBM 3380, a ccmplete revolution takes 16.8 ms. To account for the I/O rotational delay, one-half of the maximum rotational delay (8.4 ms) is used for all practical purposes (Ranade, 1987) instead of the delay time for a complete revolution. If a 4K-byte CI is transferred to or from an DASD over a 3-MBS channel, the transfer delay is given by 1.3 ms. Using the values given above, we can find the optimal reorganization point in terms of $Z_j(t)$, as well as in terms of time, from equation (6.2.9). In finding the optimal reorganization point, we experimented with four access cost deterioration rates over the different rate of retrieval and update queries where the initial database size is 50,000, and the portion of free space in CI and CA are 20% and 10%, respectively. The experiment results are given in Table 7.1.

One observation from Table 7.1 is that if the cost deterioration rate increases, the optimal reorganization time is shortened. For example, the reorganization time is shortened from 34.5 hours when $\epsilon = 0.01$ to 28.7 hours when $\epsilon = 0.04$, where $\gamma_j = 400$. This agrees with the fact that if the database is deteriorated fast, the reorganization is called frequently. The other observation is that if the query rate is increased, the optimal reorganization time is

also shortened and that if the query rate is too high, the reorganization is called even before the time of a CA split. For example, with $\epsilon = 0.02$, the optimal reorganization time is 50.5 hours when $\gamma = 100$, but it is 12.5 hours when $\gamma = 12{,}000$. However, when $\gamma = 12{,}000$, only one CA split is permitted before the reorganization is called, and, if γ is greater than 12,000, the reorganization is called even before the database experiences one CA split. A similar pattern of the optimal reorganization points, which vary depending on the values of ϵ and γ, is also observed in Maruyama and Smith (1976).

To determine the effect of free space in a CI or in a CA, experiments were done for three different loading factors and four different free CI values with the initial database size of 50,000. These are significant experiments because it is known that the free spaces within a VSAM file have contributed significantly to preventing frequent CI or CA splits. Table 7.2 shows the effect of free space combinations (% of free space in a CI; % of free space in a CA) on optimal reorganization point.

Observation of Table 7.2 indicates that the increase of free space both in a CI and in a CA contributes to lengthen the optimal reorganization point. This indicates that the free space allocations in CIs and CAs reduce the database operational costs, especially costly CA splits and reorganizations are done less frequently. One must

consider, however, that increasing the free spaces requires additional storage costs. As can be seen in Table 7.2, Z(0) increases, along with the increase in free spaces. Therefore, the free space allocation should be decided in the way of balancing the operational cost reduction and the additional storage costs. If we compare the net effect of free space increase in a CI and that in a CA, the increase of free space in a CI more significantly contributes to lengthen the reorganization period. For example, when the free space in a CI is increased 6.7% from 13.3% to 20% (where the free space in a CA is 10%), the optimal reorganization point is increased by 12.4 hours, but when the free space in a CA is increased 10% from 10% to 20% (where the free space in a CI is 13.3%), the optimal reorganization point is increased only 6.6 hours. The storage usage also shows that the increase of free spaces in a CI is more economical than that in a CA.

Table 7.1. Effects of deterioration rate and query rate on optimal reorganization points

γ \ ϵ		0.01	0.02	0.03	0.04
100	time	54.9	50.5	47.1	44.7
	$Z(t)^*$	86	78	72	68
200	time	43.1	39.9	37.3	35.7
	$Z(t)^*$	65	60	56	53
400	time	34.5	31.9	30.1	28.7
	$Z(t)^*$	51	48	45	44
1000	time	26.1	24.3	22.9	22.1
	$Z(t)^*$	41	39	38	37
2000	time	22.1	20.1	18.9	18.3
	$Z(t)^*$	37	35	35	34
4000	time	18.3	16.7	15.7	15.1
	$Z(t)^*$	34	33	33	33
8000	time	15.1	13.9	13.1	12.7
	$Z(t)^*$	33	32	32	32
10000	time	14.3	13.1	12.5	
	$Z(t)^*$	32	32	32	Z(0)
12000	time	13.7	12.5		
	$Z(t)^*$	32	32	Z(0)	Z(0)
18000	time	12.3			
	$Z(t)^*$	32	Z(0)	Z(0)	Z(0)
18000 >	time				
	$Z(t)^*$	Z(0)	Z(0)	Z(0)	Z(0)

{ Z(0) = 31, free space (CI,CA) = (20%,10%) }

Table 7.2. Free space effects on optimal reorganization points { $\epsilon = 0.02$, $\gamma = 400$ }

CA(%) \ CI(%)		13.3	20.0	26.7
10.0	Z(0) time Z(t)*	29 19.5 47	31 31.9 48	34 47.5 50
20.0	Z(0) time Z(t)*	33 26.1 50	35 41.1 51	38 59.1 53
30.0	Z(0) time Z(t)*	37 32.1 53	40 49.3 55	44 69.9 59
40.0	Z(0) time Z(t)*	43 38.5 59	47 57.9 63	51 80.7 66

Chapter 8

SUMMARY AND CONCLUSION

Efficient implementation of the physical database structure is an important issue in the design and development of database systems that grow over time. This research has addressed the issue of determining efficient physical database implementation in a VSAM file environment by considering both the physical database design and the database performance analysis that leads to the database reorganization problem.

The initial design of the physical database structure addressed in this research is a function of the estimated values of the various parameters. As implementation of the initial design proceeds and the database is reorganized, these estimates are revised and used as input to the redesign, and the cycle starts anew. This process continues in a similar manner so that the physical design of the database and the reorganization policies are, in effect, maintained at an efficient level over the life of the database.

In the micro view of the research, we are concerned with the specific details that describe how the model is actually implemented.

For the analysis of the behavior of the database in a VSAM file, two basic models for the physical database design

have been developed based on fringe analysis. The importance of the modeling is that it is the first attempt in the literature and that it gives the basis for the combined analysis of physical database design and performance analysis that leads to the issue of database reorganization in a VSAM file environment. The significance of the first model, which considers only record insertions, is that it helps to verify the correctness of the modeling. In the environment where there is no other model to be compared directly, the storage utilization calculated from this model provides a good clue for the correctness of the model. Because, as was discussed in Section 5.1, if the file life time is long enough, the behavior of the model is similar to B-tree, and the storage utilization in a B-tree is well known. The comparison of the utility values that come from the model of this research and that from the B-tree analysis gives a good match within the order of error range.

The second model, which was developed by extending the concept of the first model, considers both record insertions and deletions. In this model, a birth and death process was used to describe the transitional behavior of fringes. The expected number of CI's was obtained by the function of initial loading factor, insertion rate, deletion rate, number of total records in the system, CI capacity size, etc. One observation from the numerical experiment of this

model is that the ratios between the total number of CIs at time t and that at initial loading time are similar regardless of the CI capacity sizes of files if the initial storage utilizations are set equal. Another observation is that the total number of CIs in a file becomes almost the same regardless of the loading factor if the file life is long enough. Based on this model, further study of the physical database structure change has been conducted including CA splits. Ci and CA splits are a major cause of poor performance. While both increase I/O's at the time of the split, a CA split generates far more I/O's than a CI split, increasing run time and CPU time for a batch job and increasing response time for an on-line transaction. For this reason, the number of CAs is used as a basis of database performance analysis. If database deterioration caused through the CA splits badly prohibits the efficient database operations, database reorganization is called. Therefore, determining the optimal reorganization policy is introduced as the next issue to be discussed.

Numerical analysis performed using the equations developed in Chapter 6 shows first, the optimal reorganization point highly depends both on the database access cost deterioration rate and on the database query rate. The increase of them cause the reduction of the optimal reorganization time. Second, the larger the free space allocation, the more the optimal reorganization point

is extended. However, the balance between the operational cost reduction and the storage cost increase should be considered.

For the further development of this research, it is recommended to consider the more detailed analysis on database operations together with the cost analysis based on index structure changes.

REFERENCES

Baeza-Yates, R.A., Modeling Splits in File Structures, Acta Informatica, 26, (1989 a) 349-362.

Baeza-Yates, R.A., Expected behavior of B^+-trees under Random Insertions, Acta Informatica, 26, (1989 b) 439-471.

Chin, Y.H., An analysis of distributed free space in an operating and data management systems environment, IEEE Trans. Software Eng. SE-4, (1978) 436-440.

Chu, J.H., An analysis of B-trees and Their Variants, Information Systems, 14, 5, (1989) 359-437.

Comer, D., The Ubiquitous B-tree, ACM Computing Surveys, 11, 2, (June 1979) 121-137.

Cooper, R.B. and Solomon, M.C., The Average Time Until Bucket Overflow, ACM Trans. Database Syst. (Sept. 1984) 392-408.

Cullen, C.G., Linear Algebra and Differential Equations : An Integrated Approach, Prindle, Weber & Schmidt, Boston, (1979).

Eisenbarth, B., Ziviani, N., Gonnet, G., Mehlhorn, K., Wood, D., The Theory of Fringe Analysis and its Application to 2-3 Trees and B-Trees, Inf. Control, 55, (1982) 125-174.

Heyman, D.P., Mathematical Models of database degradation, ACM Trans. Database Syst., 7,4, (Dec. 1982) 615-631.

Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, Holden-Day Inc., Oakland, (1980) 888 pp.

IBM Corp., MVS/Extended Architecture VSAM Administration Guide, San Jose, California, (1985).

Keehn, D.G. and Lacy, J.O., VSAM data set design parameters, IBM Sys. J., 3, (1974) 186-202.

Koushik, M.V., Optimal Physical Database Design and Reorganization Policies, Ph.D. Dissertation, Univ. of Iowa, (1987) 131 pp.

Larson, P., Analysis of index-sequential files with overflow chaining, ACM Trans. Database Syst., 6, 4, (Dec. 1981) 671-680.

Leung, C.H.C., Dynamic Storage Fragmentation and File Deterioration, IEEE Trans. on Software Eng., 12, 3, (Mar. 1986) 436-441.

Martin, J., Computer Data-Base Organization, Prentice Hall, (1977) 576 pp.

Maruyama, K. and Smith, S.E., Optimal Reorganization of Distributed Space Disk Files, Comm. ACM (Nov. 1976) 634-642.

Mendelson, H. and Yechiali, U., Optimal Policies for Database Reorganization, Operations Research, 29, 1, (Jan. 1981) 23-36.

Park, J.S., Bartoszynski, R., Prabuddha De and Pirkul, H., Optimal Reorganization Policies for Stationary and Evolutionary Databases, 36, Mgt. Sci. (May 1990), p.613

Pritsker, A.A.B., Introduction to simulation and SLAM II, Halsted Press Book, John Wiley & Sons, New York, (1986) 837 pp.

Quitzow, K.H. and Klopprogge M.R., Space Utilization and Access Path Length in B-trees, Information Systems, 5, (1980) 7-16.

Ranade, J., VSAM : Performance, Design, and Fine Tuning, Macmillan Publishing Company, New York, (1987) 280 pp.

Ranade, J. and Ranade H., VSAM : Concepts, Programming, and Design, Macmillan Publishing Company, New York, (1986) 358 pp.

Ross, S., A First Course in Probability, Macmillan Publishing Company, New York, (1984) 392 pp.

Severance, D.G. and Duhne, R., A Practitioner's guide to addressing algorithms, Comm. ACM, 19, 6, (1976) 314-326.

Shneiderman, B., Optimum database reorganization points, Comm. ACM, 16,6, (June 1973) 362-365.

Sockut, G. and Goldberg, R., Database reorganization - Principles and practice, ACM Computing Survey, 11, 4, (Dec. 1979) 371-396.

Teorey, T.Y. and Fry, J.P., Design of Database Structures, Prentice-Hall, (1982) 492 pp.

Tuel, W.G., Optimum reorganization points for linearly growing files, ACM Trans. Database Syst., 3, 1, (Mar. 1978) 32-40.

Van der Pool, J.A., Optimum Storage Allocation for a file in Steady State, IBM J. Res. Dev., (1973) 27-38.

Wiederhold, G., File Organization for Database Design, McGraw-Hill, New York, (1987) 619 pp.

Yao, S.B., Das, K.S., and Teorey, T.J., A dynamic database reorganization algorithm, ACM Trans. Database Syst., 1, 2, (June 1976) 159-174.

Yao, A., On random 2-3 trees, Acta Informatica, 9, (1978) 159-170.

VITA

SUNG-EON KIM

Birth Date : March 14, 1953

EDUCATION :

1988-Pre	Ph.D. student in Quantitative Business Analysis Major Area : Management Information Systems Minor Area : Computer Science Louisiana State University
1985-1988	M.S. in Systems Science Louisiana State University
1982-1984	M.S. in Physical Oceanography Florida State University Tallahassee, Florida
1973-1977	B.S. in Oceanography Seoul National University Seoul, Korea

ACADEMIC/PROFESSIONAL EXPERIENCES :

1990-Pre	Teaching Assistant Dept. of Quantitative Business Analysis Louisiana State University
1988-1990	Graduate Assistant Dept. of Quantitative Business Analysis Louisiana State University
1987-1988	Programmer Dept. of Civil Engineering Louisiana State University
1986-1987	Programmer Ports and Waterways Institute Louisiana State University
1982-1984	Research Assistant Dept. of Oceanography Florida State University
1980-1982	Researcher Korea Research Institute for Human Settlements Seoul, Korea
1978-1980	Researcher Agency for Defense Development Chinhae, Korea

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Sung-Eon Kim

Major Field: Business Administration
(Quantitative Business Analysis)

Title of Dissertation: Modeling of Physical Database Design and Performance Analysis with Emphasis on VSAM files

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

June 24, 1991